Image Preprocessing Strategies for Improving ArUco Marker Detection Underwater

BELIER Titouan
3rd year engineering student
ENSTA Bretagne
29200, Brest, FRANCE
titouan.belier@ensta-bretagne.org

***Abstract*—In underwater computer vision, enhancing the detection of ArUco markers presents significant challenges due to environmental factors such as turbidity and variable lighting conditions. This study explores image preprocessing strategies aimed at improving ArUco marker detection performance in underwater environments. Three distinct preprocessing approaches are investigated: an enhancements to the Adaptive Histogram Equalization, a frequency-domain-based method focusing on local and global processing, and an automatic preprocessing technique targeting lighting correction, noise reduction, contrast enhancement, and color adjustment. Experimental validation is conducted to evaluate the effectiveness of these strategies using real underwater imagery. This research sheds light on effective preprocessing techniques tailored for underwater ArUco marker detection applications, offering insights for the development of robust underwater computer vision systems.**

***Index Terms*—preprocessing, vision, aruco, underwater, ROV**

I. Introduction

In the scope of an other project, we are attempting to dock a BlueRobotics BlueROV robot [5] to an appendage. To achieve this docking, we employ vision-guided techniques. This vision guidance is facilitated using ArUco markers, allowing us to position ourselves in space relative to the appendage. By knowing the distance between the ArUco markers and the appendage, we can guide our robot to successfully accomplish the docking maneuver. Despite significant advancements in underwater exploration, challenges persist in underwater image and video processing techniques, particularly in computer vision applications. One of the primary challenges stems from light absorption and scattering in water, leading to color shifts and contrast degradation in underwater images. Color distortion, often appearing bluish-green, arises from the propagation of light wavelengths in water, while contrast degradation results from the random attenuation and scattering of light by suspended particles in water.

To address these issues and enhance the quality of underwater imaging, various image restoration and enhancement methods have been proposed. These methods aim to recover degraded images by modeling the degradation process directly. In this context, our study explores the work of three papers that contribute to improving underwater detection. The paper Pizer, Amburn and Austin (1987) [3], proposes a novel algorithm that discusses enhancements to the Adaptive Histogram Equalization (AHE) method to improve its speed and quality. It introduces variants such as Interpolated Adaptive Equalization, Weighted Equalization, and Thresholded Equalization to mitigate noise amplification. Another paper presents an automatic preprocessing method tailored for underwater image denoising, focusing on lighting correction, noise reduction, contrast enhancement, and color adjustment Bazeille (2008) [2]. Lastly, Donghui and Wanchun and Xiaogang (2017) [1] introduces a two-step approach for single underwater image enhancement, addressing color distortion and low contrast issues separately to achieve optimal results.

Through our analysis and comparison of these approaches, we aim to evaluate their effectiveness in enhancing ArUco marker detection underwater, thereby contributing to the development of robust underwater computer vision systems. A comparative analysis of these three methods in this specific context would determine which one is best suited to the needs of the ROV system for effective underwater marker detection.

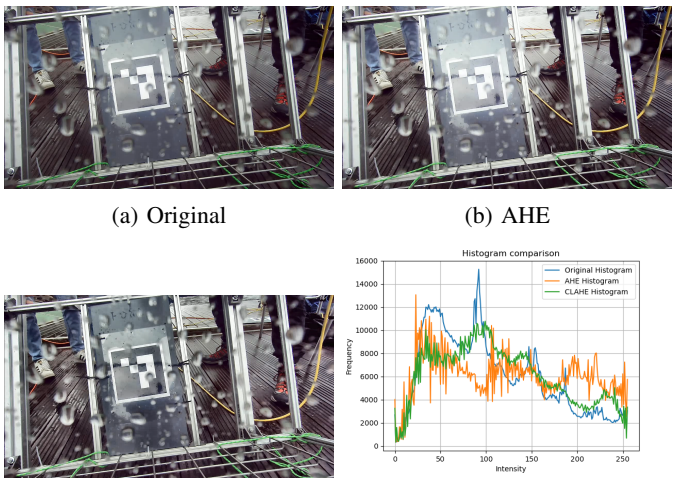
II. Contrast-limited adaptive histogram equalization (CLAHE)

The provided paper explores the field of image restoration and presents an innovative method to enhance the visual quality of degraded images. This study originates from a research team in the Computer Science department at Stanford University, known for its pioneering work in computer vision and image processing.

The primary focus of this paper lies in proposing an advanced algorithm for contrast histogram equalization, which constitutes a crucial step in the image restoration process. This technique corrects brightness variations and enhances image contrast, resulting in a significant improvement in visual quality and detail perception.

To design their contrast histogram equalization algorithm, the researchers leveraged advanced techniques in image processing and computer vision. They began by analyzing the luminance histogram of each image to be restored to identify areas with significant contrast variations. The Figure represents the CIELAB color space. The world is divided into three dimensions: the luminance dimension along the vertical axis, the red-green axis, and the yellow-blue axis. The CIELAB color representation offers a standardized and perceptually uniform color space, facilitating accurate color analysis and manipulation in various imaging applications [4]. Afterward,

they developed an adaptive transformation function based on these local contrast variations, allowing precise adjustment of the distribution of grayscale levels in each region of the image.



(a) Original (b) AHE

(c) CLAHE (d)

Fig. 1: Comparison AHE vs CLAHE

CLAHE (Contrast Limited Adaptive Histogram Equalization) adds a local contrast limitation to AHE (Adaptive Histogram Equalization), allowing for better control over the intensity distribution and preventing excessive amplification of noise in low-contrast regions of the image. The Figure 1 represents a comparison of the AHE and CLAHE methode, in Figure 1b you can see the difference between the two algorithm thanks to a diagram.

The innovative approach of this paper offers several advantages over traditional histogram equalization methods. By considering local contrast variations, the algorithm proposed by the researchers can preserve fine details while improving the overall contrast of the image. Moreover, its adaptive nature enables it to automatically adapt to different types of images and scenes, making it particularly effective in diverse contexts, ranging from the restoration of old images to the enhancement of the quality of modern digital images.

III. Two step enhancing strategy

The paper titled "Two-Step Approach for Single Underwater Image Enhancement" presented at the 2017 International Symposium on Intelligent Signal Processing and Communication Systems in Xiamen, China, addresses the challenges of improving the quality of underwater images, which often suffer from color distortion and low contrast due to light absorption and scattering in water.

The paper highlights various existing methods for underwater image enhancement, including those focusing on contrast distortions, blurring effects, polarization haze removal, wavelength compensation, and restoration based on dictionary learning and radiation transfer function. These methods, while effective to some extent, have limitations such as sensitivity to modeling assumptions and computational complexity.

In response to these challenges, the authors propose a two-step enhancement strategy that aims to individually address color distortion and low contrast issues in underwater images.

A. Color Correction

The first step involves a color correcting strategy based on piecewise linear transformation to mitigate color distortion caused by light absorption in water. This method is robust and effectively deals with varying degrees of color distortion.

Firstly, they operate under the assumption of a grey world, meaning that the average value of the image is equal to 128. Then, they employ two methods to correct the color. The first method involves selecting, for each RGB channel, the average of the values as a criterion for correcting the image.

$$S_{c_{\text{CR}}} = \begin{cases} (S_c - S_{c_{\text{mean}}}) \times \frac{S_{c_{\text{min}}} - 128}{S_{c_{\text{min}}} - S_{c_{\text{mean}}}} + 128, & \text{si } S_{c_{\text{mean}}} \leq 128 \\ (S_c - S_{c_{\text{mean}}}) \times \frac{S_{c_{\text{max}}} - 128}{S_{c_{\text{max}}} - S_{c_{\text{mean}}}} + 128, & \text{si } S_{c_{\text{mean}}} > 128 \end{cases}$$

where $c \in \{R, G, B\}$. $S_{c_{\text{mean}}}$, $S_{c_{\text{max}}}$, and $S_{c_{\text{min}}}$ are the maximum and minimum in the c channel, respectively, and SCR is the color corrected image.

However, this method is not always satisfactory, particularly underwater where red wavelengths are quickly attenuated.

$$S_{c_{\text{CR}}} = \begin{cases} (S_c - S_{c_{\text{mean}}}) \times \frac{S_{c_{\text{min}}} - 128}{S_{c_{\text{min}}} - S_{c_{\text{mean}}}} + 128, & \text{si } S_{c_{\text{mean}}} \leq 128 \\ (S_c - S_{c_{\text{mean}}}) \times \frac{S_{c_{\text{max}}} - 128}{S_{c_{\text{max}}} - S_{c_{\text{mean}}}} + 128, & \text{si } S_{c_{\text{mean}}} > 128 \\ S_c - \lambda \times (S_{c_{\text{mean}}} - 128), & \text{si } P_c < 0.7 \end{cases}$$

where λ is a positive parameter to control the shifting range and P is the probability of pixel values that are less than or equal to 40.

B. Optimal constrast improvement

The second step focuses on enhancing contrast in underwater images to highlight objects and details. A novel optimal contrast method is proposed, which efficiently reduces artifacts and improves contrast without requiring prior knowledge of imaging conditions.

The proposed approach is straightforward to implement as it uses CLAHE algorithm as explained in former paper. Experimental results demonstrate significant improvements in color, contrast, naturalness, and object prominence in enhanced underwater images.

The paper concludes that the proposed two-step enhancement strategy offers an effective and efficient solution for improving the quality of single underwater images, making it suitable for real-time applications. Additionally, the computational time is reasonable, indicating its practical feasibility.

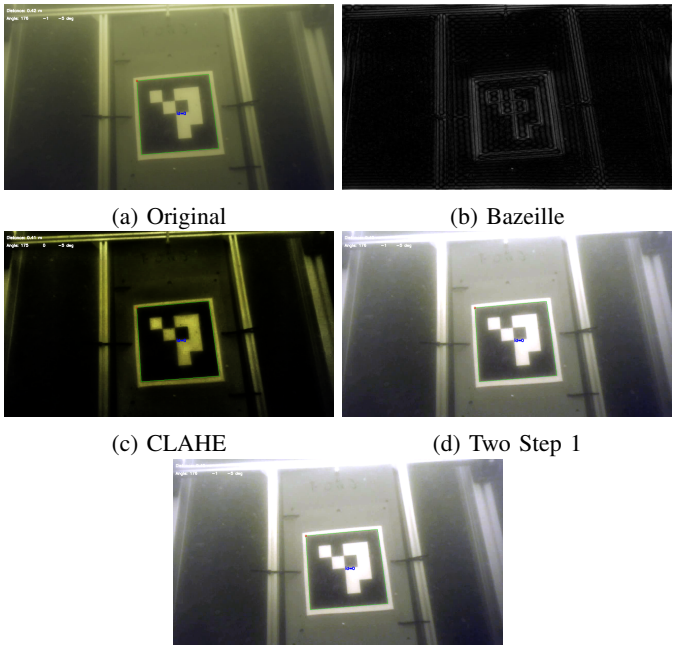
IV. Algorithm comparison for Aruco's detection

To compare the relevance of these preprocessing algorithms in the context of Aruco detection, two aspects can be compared. Firstly, the 'Detection Rate', which is the percentage of algorithm detection over a large number of images. The second aspect is 'Temporal'. Indeed, these algorithms will then be embedded on autonomous underwater robots such as ROVs. In the case of a robot aiming to orient itself in space

using Arucos, it is important to ensure that the preprocessing does not cause significant delays in the robot's guidance. After conducting a comparative analysis of these two models, we will create a decision matrix to choose the most relevant algorithm according to us.

We will use two videos from a BlueRov at a distance of approximately 50 cm from a 15 cm long Aruco 4x4. The robot is located in Lake Guerlédan, in central Brittany - France. The water in the lake is very murky, so we face all the problems of underwater vision: water turbidity, a light gradient between the surface and deeper water, water bubbles, etc. These complications related to the aquatic environment cause distortion of vision and chromatic aberrations.

You can find these two videos here: Vidéo 1 and Vidéo 2. In the first video, the camera is first in the air and then we immerse the robot underwater. This video is interesting to study the differences in detection between air and water. We will choose the robust algorithm that allows us good detection in both cases. The second video is another underwater video, which we chose to have a second example of underwater video. The results on both coupled videos will allow us to make our choice.



(a) Original (b) Bazeille

(c) CLAHE (d) Two Step 1

(e) Two Steps 2

Fig. 2: Different algorithm results

The Figure 2 illustrates the diverse result images generated by the algorithms. In Figure 2a, the original view of the robot is presented, displaying a noticeable light gradient and a color tone shifting towards green.

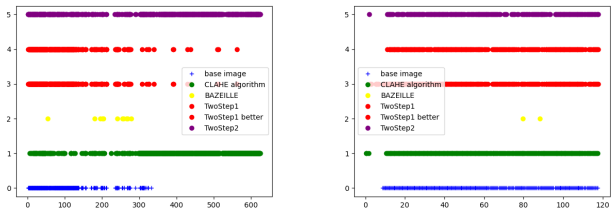
Figure 2b showcases the output of the algorithm derived from Bazeille's paper. While the edges are clearly defined, the loss of flat colors disrupts the detection process. Additionally, this algorithm exhibits significantly longer detection times compared to others, as later observations will reveal.

In Figure 2c, the original image is subjected to Contrast Limited Adaptive Histogram Equalization (CLAHE). Although the image may not appear visually appealing and might even seem deteriorated, it demonstrates exceptional performance in Aruco detection.

Lastly, Figures 2d and 2e portray the output of Donghui and Wanchun's algorithm at two processing stages. Despite minimal observable differences between the images, the resulting outputs exhibit the sharpest details among all algorithms, achieving impeccable image restoration. However, this algorithm does not necessarily offer the best detection performance as we will see after.

A. *Detection Rate Aspect*

To conduct our detection rate analysis, at each time step, we capture an image, apply preprocessing to it, and measure the elapsed time. For this study, we use a laptop with an Intel® Core™ i5-6300U CPU @ 2.40GHz × 4 processor. To compare these algorithms, we use a common reference. In our case, it's a video taken from the BlueRov's perspective in Lake Guerlédan. Knowing the actual duration of the video, we can compare the time taken by the program to retrieve the video, perform preprocessing, and conduct detection. We will display all the results in a diagram to compare our findings.



(a) Video 1 (b) Video 2

Fig. 3: Comparaison de détection d'Aruco

The Figure 3a is a representation of the detection. At each frame, the code will look if there is a detection. If it is the case, it will add a point. This way, if there is a hole in the line made by all points, we can monitor that the algorithm don't allow the ROV to detect the Aruco. The y axis is only to have every detection algorithm results in one diagram. As you can see, the results of Bazeille's algorithm are not satisfying as our implementation only enhance edges and not only contrast. There are only few detections made. On the contrary, the CLAHE algorithm detect really often, the two-step algorithm is also pretty accurate.

To quantify the dectection rate, we have detailed in Table I the detection rates for each algorithm and each video to ensure a fair comparison. It can be observed that Aruco detection is consistently better with a simple CLAHE. Video 1 yields poorer results compared to the second one because the detection was more challenging, especially during the phase when the robot is submerged underwater, resulting in a prolonged period with no possible detection. Henceforth in the

report, we will not discuss the results of Bazeille, as they are unusable for Aruco detection, presumably due to our Python implementation being incorrect.

Method	Detection rate (Video 1)	Detection rate (Video 2)
Original	0.265	0.920
Clahe	0.614	0.920
Bazeille	0.009	0.008
Two Step 1	0.235	0.849
Two Step 1 Better	0.235	0.807
Two Step 2	0.537	0.849

TABLE I: Detection rates for different methods on Video 1 and Video 2.

Finally, the better algorithm for detection is CLAHE as it really improves the detection rate. The most interesting result is for the two step algorithm. In fact, the resulting image of this algorithm seems really better when we look at it, this algorithm is very good to create good-looking images. On the contrary, it's not the perfect algorithm for Aruco's detection as it keeps the halo from the above light-source.

B. Temporal Aspect

To study the temporal aspect of these algorithms, we record the time elapsed since the beginning of the video for each frame. An algorithm will be considered relevant if the image processing time allows for real-time analysis without loss of information. However, we observe that some image processing algorithms are computationally expensive. In such cases, the frames per second (fps) drop drastically, and the robot cannot use all the images for its guidance.

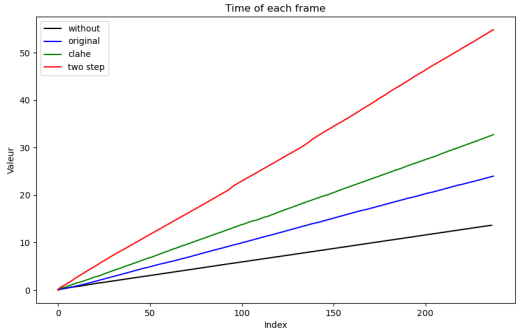


Fig. 4: Time Comparison

As the time gap remains constant regardless of the video duration, we chose to conduct this study on a 14-second video. Figure 4 illustrates the time elapsed since the beginning of the video for each frame. It can be observed from Figure 4 that the algorithm by Donghui and Wanchun and Xiaogang (2017) [1] causes the most processing delay. The average duration for viewing the video and performing detection alone is 24 seconds. Adding CLAHE preprocessing results in a total duration of 32 seconds, while adding TwoStep preprocessing yields a total duration of 54 seconds.

These results are significant and greatly affect the robot's vision-guided navigation. Indeed, our robot may lose up to 74% of the information required for navigation, as shown in Table II.

	Analysis duration	**FPS**	**Information loss (%)**
Video Only	14	60	0,00%
Simple detection	24	35	41,67%
CLAHE + detection	32	26	56,25%
TwoStep + detection	54	16	74,07%

TABLE II: Summary of analysis results

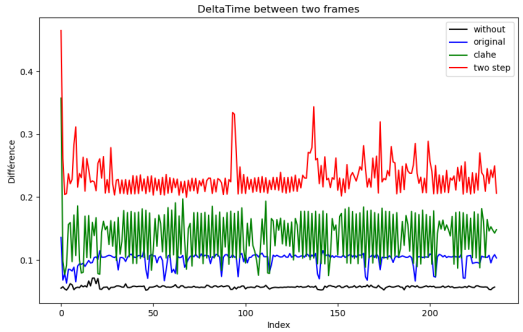


Fig. 5: Time comparison between different algorithms

The Figure 5 represents the time difference between two frames of the video at each frame. This graph allows us to better analyze the reasons for these slowdowns. We find that the twoStep algorithm is slower than the others and often experiences peaks of delay, suggesting that it may struggle more under certain conditions to perform the processing. However, we can now observe that the CLAHE algorithm shows less consistency in its processing time, as there is a 60 ms difference between the best and worst detections, whereas the TwoStep algorithm has a difference of only 20 ms.

Additionally, it is noticeable that at the beginning of the video, there is a significant time gap between each of the first frames for each of the algorithms. This is simply because the initial frames are affected by the camera startup, which produces inconsistent images. Therefore, these initial frames should not be considered in the analysis.

V. Conclusion

In conclusion, this study has evaluated the effectiveness of various image preprocessing strategies to enhance ArUco marker detection in underwater environments. The results obtained reveal that the examined preprocessing methods, notably Contrast Limited Adaptive Histogram Equalization (CLAHE) and the two-step approach proposed by Donghui et al., play a crucial role in improving the visual quality of underwater images, thereby facilitating more accurate detection of ArUco markers.

The comparative analysis demonstrated that CLAHE preprocessing offers significant improvements in terms of visual quality and contrast, leading to enhanced marker detection accuracy. On the other hand, while the two-step approach also showed promising results in terms of image quality, it is not

really suited for Aruco's detection and it presents drawbacks in terms of processing time, which may limit its utility in real-time detection applications.

Considering these findings, it appears that CLAHE preprocessing is better suited for ArUco marker detection applications in underwater environments, providing an optimal balance between image quality and processing time. However, future work could involve further optimization of the parameters of these preprocessing methods or exploration of other techniques that are faster while maintaining a high level of detection accuracy.

Additionally, an idea for future development would be to migrate the preprocessing algorithms to more efficient programming languages such as C++, which would significantly improve processing times and pave the way for more robust and efficient real-time applications.

REFERENCES

[1] Wei, Donghui and Chen, Wanchun and Chen, Xiaogang. (2017). A Two-Step Approach for Underwater Image Enhancement. 985-988. 10.1109/ICCSEC.2017.8446772.

[2] Bazeille, Stéphane and Quidu, Isabelle and Jaulin, Luc and Malkasse, Jean-Philippe. (2008). Une méthode de pré-traitement automatique pour le débruitage des images sous-marines.

[3] S. M. Pizer, E. P. Amburn, J. D. Austin, and et al, "Adaptive histogram equalization and its variations," Computer vision, graphics, and image processing, vol. 39, no. 3, pp. 355–368, 1987

[4] Ly, Bao and Dyer, Ethan and Feig, Jessica and Chien, Anna and Bino, Sandra. (2020). Research Techniques Made Simple: Cutaneous Colorimetry: A Reliable Technique for Objective Skin Color Measurement. The Journal of investigative dermatology. 140. 3-12.e1. 10.1016/j.jid.2019.11.003.

[5] https://bluerobotics.com/, consulted on march 1st 2024

Solving a pentacube puzzle as an exact cover problem

Victor BELLOT
ENSTA Bretagne
victor.bellot@ensta-bretagne.org

Abstract

This article introduces the exact cover problem, some of its well known algorithmic solutions and its application domains. A practical use of the Dancing Links X algorithm [2] is proposed to solve a pentacube puzzle.

Introduction

Let's consider the two following set problems (inspired by [5]) :

set cover Given a collection $\mathcal{C}$ of subsets of a finite set Ω, a set cover is a subcollection $\mathcal{S} \subseteq \mathcal{C}$ such that $\bigcup_{S_k \in \mathcal{S}} S_k = \Omega$, and the minimum set cover problem (**SC**) is to find a cover of minimum size.

set packing Given a collection $\mathcal{C}$ of finite sets, a packing is a subcollection $\mathcal{S} \subseteq \mathcal{C}$, all members of which are mutually disjoint and the maximum set packing problem (**SP**) is to find a packing of maximum size.

Both constraint satisfaction problems can be expressed as integer linear programs of the form :

SC Minimize $b^{\mathrm{T}}x$ subject to $A^{\mathrm{T}}x \geq b$.

SP Maximize $b^{\mathrm{T}}x$ subject to $A^{\mathrm{T}}x \leq b$.

With $x \in \mathbb{F}_2^{|\mathcal{C}|}$ the variable of the program (encoding the subcollection $\mathcal{S}$), b a vector full of ones of size $|\mathcal{C}|$ and $A \in \mathcal{M}_{|\mathcal{C}|,|\Omega|}(\mathbb{F}_2)$ the constraint matrix (encoding the collection $\mathcal{C}$).

The exact cover (**EC**) problem can therefore be defined as the intersection of **SC** and **SP**. A solution of **EC** covers Ω while having all its members mutually disjoints. Equivalently, an exact cover of Ω is a subcollection $\mathcal{S}$ of $\mathcal{C}$ that partitions Ω.

No matter how it is stated, **EC** has been proven to be NP-complete :

- it's NP : candidate solutions can be verified in polynomial-time
- it's NP-hard : any NP problem can be translated into **EC** using polynomial-time reduction

Unless P (the set of problems that can be solved in polynomial-time) equals NP, **EC** can't be solved easily (in polynomial-time).

Classical algorithmic solutions

Approximation

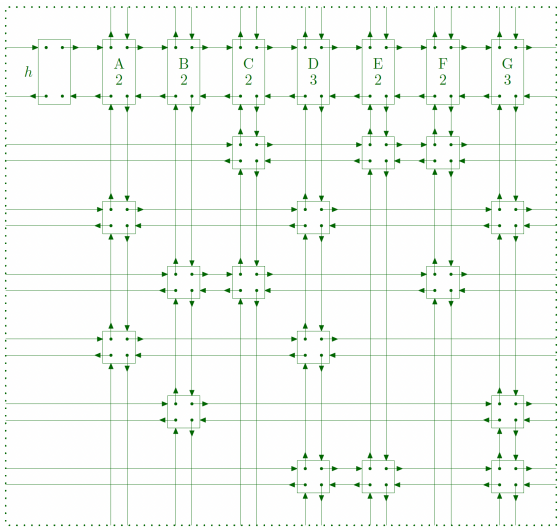
Because of the hardness of **EC**, a good idea to solve such problem is to find suboptimal solutions using an accuracy/time trade-off. A survey comparing approximately optimal solutions to some covering and packing problems has been conducted in [5].

Backtracking & Dancing Links

One common approach for solving constraint satisfaction problems is the backtracking method. In the case of **EC**, we may want to enumerate every solution of the problem. A backtracking algorithm is a method for systematically exploring all possible solutions by recursively trying different options, and backtracking when a dead end is reached. The key idea is to incrementally build a solution candidate and explore different choices at each step. If a choice leads to a dead end, the algorithm backtracks to the previous decision point and tries another option. This process continues until all possible solutions have been found.

An evaluation of the performance of such method is proposed in [3]. The authors analyze the tree search done by backtracking methods (the number of nodes visited, the evolution of depth...).

One of the most famous algorithm using this method is Donald Knuth's Algorithm X [2]. In addition to the use of backtracking, Knuth introduces in his article a new data structure : dancing links. As depicted in 1a, dancing links is a doubly linked list of nodes enabling easy removing and restoring elements from the structure. It's a powerful data structure for backtracking implementation because tree exploration (forward and backward) can easily be done by removing and restoring elements of a partial solution.



(a) The Dancing Links data structure

```
1 If A is empty, the problem is solved.
2 Otherwise choose a column, c. (dtm)
3 Choose a row, r, such that A[r,c]=1. (ndtm)
4 Include r in the partial solution.
5 For each j such that A[r,j]=1,
6     delete column j from matrix A;
7     for each i such that A[i,j]=1,
8         delete row i from matrix A.
9 Recursively call this algorithm on the new A.
```

(b) Algorithm X

Figure 1: Dancing Links enable Algorithm X

Listing 1b is a pseudocode implementation of Knuth's Algorithm X. The matrix A is the constraint matrix introduced previously. On line 2, a deterministic choose of an element of Ω is made (it doesn't matter how it is done as long as it is done consistently). On line 3, a non-deterministic choose of an element of $\mathcal{C}$ is made. It is where the tree exploration happens and where backtracking is done (in the case where no row r satisfies $A[r, c] = 1$).

Satisfiability

Another well known NP-complete problem is the satisfiability (SAT) problem. The aim is to find the interpretations that satisfy a given boolean formula. In other words, it asks how can the variables of a given boolean formula be consistently replaced by the values TRUE or FALSE in such a way that the formula evaluates to TRUE.

Because it is a very fundamental problem, a lot of algorithms have been developed to solve it efficiently. The paper [1] explains how to translate **EC** into SAT, thus enabling the use of the basic Davis-Putnam-Logemann-Loveland procedure (a classical and powerful SAT solver technic). Moreover, an empirical comparison of this method with Knuth's Algorithm X is proposed.

Applications

As mention above, every problem those solutions can be checked in polynomial-time can be translated into **EC**. Therefor the exact cover problem has a wide range of application domain.

As mention in [6], **EC** has been used to formulate a variety of practical problems in such areas as capital budgeting, crew scheduling, cutting stock, facilities location, graphs and networks, manufacturing, personnel scheduling, vehicle routing and timetable scheduling among others.

A detailed implementation for military timetabling and trainee assignment problems can be found in [4]. Scheduling is a serious subject in computer science. Efficient algorithms are required to schedule processes at the OS level, to manage energy nationwide, to regulate public transportation...

Solving a pentacube puzzle

Now that we understand how to solve the exact cover problem, let's try to create a program finding the solution of a puzzle. The puzzle is made of 25 y-pentacubes that have to fit a 5x5x5 cube :

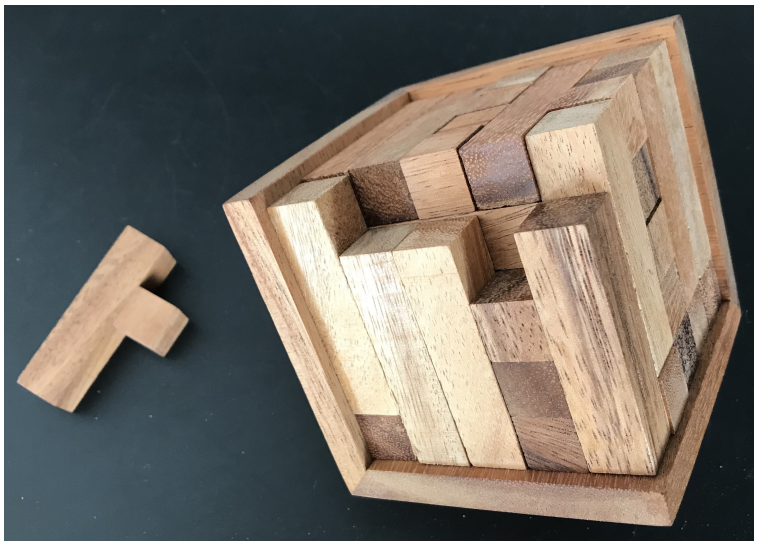


Figure 2: A 5x5x5 wooden pentacube puzzle

Puzzle 2 involves paving the 3D space using identical 3D pieces. It can naturally be expressed as an exact cover problem : Ω represents the set of all cubes composing the 5x5x5 puzzle and $\mathcal{C}$ lists all the possible ways of placing y-pentacubes in the puzzle (thus described as subsets of Ω). Not too formally :

$$\Omega = [\![1;5]\!]^3 \qquad \mathcal{C} = \{p \in \Omega^5 \mid p \text{ embodies the placement of a y-pentacube in the puzzle}\}$$

Now that our puzzle is translated into **EC**, we can create the corresponding constraint matrix A, and apply Algorithm X to find all its solutions! (see video demonstration of a Python implementation)

Conclusion

Exact cover is an easy problem to state, but quite difficult to solve. In this article we discuss some of its classical algorithmic solutions while focusing on Knuth's Algorithm X to solve a 3D puzzle. Applications and their references have been introduced to strengthen the power of this single NP-complete problem.

List of Figures

References

[1] Tommi Junttila and Petteri Kaski. "Exact Cover via Satisfiability: An Empirical Study". In: *Principles and Practice of Constraint Programming – CP 2010*. Ed. by David Cohen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 297–304. ISBN: 978-3-642-15396-9.

[2] Donald E. Knuth. *Dancing links*. 2000. arXiv: cs/0011047 [cs.DS].

[3] Grzegorz Kondrak and Peter van Beek. "A theoretical evaluation of selected backtracking algorithms". In: *Artificial Intelligence* 89.1 (1997), pp. 365–387. ISSN: 0004-3702. DOI: https://doi.org/10.1016/S0004-3702(96)00027-6. URL: https://www.sciencedirect.com/science/article/pii/S0004370296000276.

[4] Vivian Nguyen et al. "An efficient and exact algorithm for military timetabling and trainee assignment problems". In: *Computers and Industrial Engineering* 169 (2022), p. 108192. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2022.108192. URL: https://www.sciencedirect.com/science/article/pii/S0360835222002625.

[5] Vangelis T. Paschos. "A survey of approximately optimal solutions to some covering and packing problems". In: *ACM Comput. Surv.* 29.2 (June 1997), pp. 171–209. ISSN: 0360-0300. DOI: 10.1145/254180.254190. URL: https://doi.org/10.1145/254180.254190.

[6] R. R. Vemuganti. "Applications of Set Covering, Set Packing and Set Partitioning Models: A Survey". In: *Handbook of Combinatorial Optimization: Volume1–3*. Ed. by Ding-Zhu Du and Panos M. Pardalos. Boston, MA: Springer US, 1998, pp. 573–746. ISBN: 978-1-4613-0303-9. DOI: 10.1007/978-1-4613-0303-9_9. URL: https://doi.org/10.1007/978-1-4613-0303-9_9.

A Survey on 3D Reconstruction from Multiple View Images

Léo BERNARD[1]
[1]Ecole Nationale Supérieur des Techniques Avancées Bretagne (ENSTA Bretagne), Brest, Bretagne 29200, FRANCE

The abstract goes here.

***Index Terms*—Keywords : Computer vision, Stereovision, Multi-view images, 3D reconstruction.**

I. Introduction

STEREOPSIS, the ability of humans and animals to perceive depth in three dimensions, has always been a fascinating aspect of visual perception. The intricate interplay between our eyes, each capturing a slightly different perspective, allowing us to navigate the world with depth and precision. This natural phenomenon has served as a cornerstone for our understanding of spatial relationships and has proven invaluable for survival in the natural world.

In the realm of technology, the quest to replicate and harness stereopsis has led to significant breakthroughs. Computers, once limited to two-dimensional representations of the world, have now evolved to perceive depth through stereo vision. This technological leap has opened new frontiers in fields such as computer vision, enabling machines to interpret and interact with the environment in ways that were once solely within the domain of living organisms.

Traditionally, 3D reconstruction has been a meticulous process, often relying on stereoscopic pairs of images captured by calibrated cameras. These pairs provide the necessary depth cues for algorithms to triangulate and reconstruct a three-dimensional scene. Classical methods involve techniques like Structure from Motion (SfM), Multiple View Stereo (MVS) and stereo matching, where correspondences between image features are identified and used to derive the spatial structure of the scene. While effective, these classical approaches often necessitate controlled environments, precise camera calibration, and extensive computational resources.

However, the scope of stereo vision has expanded even further in the age of the Internet. With the proliferation of digital imagery on the web, computers have gained access to an extensive repository of visual data. Now, not only can computers perceive depth in individual images, but they can also reconstruct three-dimensional representations of entire cityscapes. The abundance of pictures available online has transformed the way machines "see," paving the way for the exploration and understanding of vast and complex environments.

The advent of the Internet has revolutionized 3D reconstruction by introducing an unprecedented volume of diverse visual data. The abundance of images available online enables the application of novel techniques, such as large-scale image-based reconstruction. Internet-derived images, captured by various cameras with uncalibrated parameters, provide a wealth of perspectives on a given subject, allowing for more robust and detailed reconstructions. This shift from controlled laboratory conditions to the dynamic, uncontrolled realm of the Internet has expanded the possibilities of 3D reconstruction, making it applicable to a wide array of real-world scenarios.

Despite the remarkable progress in 3D reconstruction, challenges persist in achieving accurate and reliable results. Variability in lighting conditions, occlusions, and the uncalibrated nature of images from diverse sources on the Internet can impede the accurate extraction of depth information. Additionally, the sheer volume and diversity of Internet-acquired images pose challenges in terms of data management and processing efficiency. Ensuring the alignment of multiple views, handling noisy data, and addressing scalability issues become crucial aspects in the pursuit of high-fidelity 3D reconstructions. Overcoming these challenges, especially in the context of uncalibrated stereo vision from the Internet, is essential for the continued advancement and practical application of 3D reconstruction techniques in complex, real-world environments.

In this survey, we delve into the fascinating realm of 3D reconstruction from multiple view images, exploring the technological advancements that enable computers to navigate the rich tapestry of our three-dimensional world, as seen through the lens of the Internet.

II. Establishing correspondences between multiple images

In the pursuit of computing the 3D position of an observed point in an image, the first step is to get different views of this point. This is the problem of correspondence : finding correspondences between different views of the same points in a set of images. This is a well studied subject in computer vision, it relies on robust techniques for feature matching and geometric refinement. One pivotal approach involves the utilization of the Scale-Invariant Feature Transform (SIFT) and the Random Sample Consensus (RANSAC) algorithm.

SIFT is a method developped by Lowe in 1999 [1] to extract features from an image. This algorithm stands out as a powerful tool for finding correspondences due to its ability to detect and extract distinctive features invariant to scaling, rotation, translation, and changes in illumination. By focusing

on keypoint detection, localization, orientation assignment SIFT provides a set of robust features that serve as key landmarks across images. These distinctive features, essential for matching, contribute to the establishment of correspondences between different views. Despite its local nature, SIFT forms a crucial foundation for subsequent analysis in computer vision applications.

Using those features, we can match some points in two different images (if the features are close).

However, SIFT features are local and lack global information about the image. To address this, the Random Sample Consensus (RANSAC) algorithm is employed to impose geometric constraints on the matching process. RANSAC deals with outliers in the data and iteratively estimates model parameters by randomly selecting minimal subsets of feature points. More precisely, RANSAC method estimates the best fondamental matrix (which is the matrix that contains the parameters of the camera) for the current points. This matrix can then be used to project a point seen from a camera into another image from another camera, using the following equation, we get a line (called epiline) where the point should be in the other image.

$$x_{ij}^{\top} F x_{ij} = 0. \tag{1}$$

RANSAC method then excludes the outliers : the points that are too far from the epiline where they should be. Then it iterates the two previous steps to improve the estimation of the fondamental matrix and reject more outliers.

This technique is particularly useful in refining matches based on SIFT features, enhancing the accuracy of correspondence found between our images.

The next fig(1) represents the results of matching of features in two images using SIFT to extract the feautures then RANSAC to refine the matches.

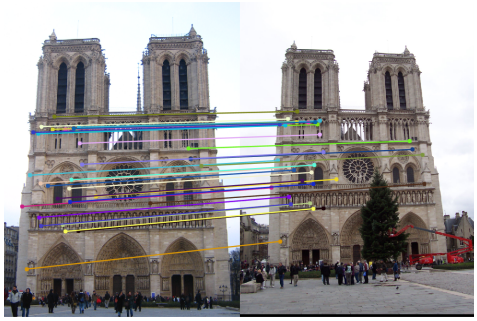


Fig. 1. Matching between keypoints found using Sift then RANSAC

III. 3D RECONSTRUCTION OF AN ENVIRONMENT

A. Structure from Motion (SfM)

Structure from Motion is a computer vision technique that addresses the challenge of reconstructing the three-dimensional (3D) structure of a scene from a set of two-dimensional (2D) images. The fundamental difficulty lies in the fact that a photograph captures a 3D world as a flat, 2D projection, causing a loss of depth information. SfM aims to recover this lost depth by leveraging multiple images of a scene.

By establishing correspondences between the 2D projections, we process a triangulation of the 3D point across different images and we gain valuable constraints on the position of the point in 3D and the parameters of the camera (position and intrinsic parameters).

In the context of SfM, the process involves iteratively refining estimates for the 3D positions of points and camera parameters to minimize the difference between predicted and observed 2D projections (called reprojection error). Mathematically, this is formulated as an optimization problem aiming to minimize the squared reprojection error (see fig2).

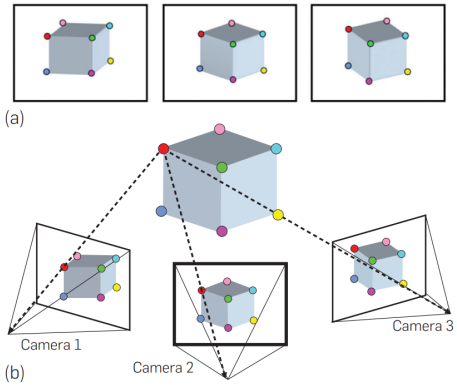


Fig. 2. Reprojection Error on 3 views of a cube [5] :
The triangulation based on the images from camera 1 and camera 2 gives a 3D position to the angle observed. This point is projected into camera 3 using its supposed position and parameters, then this projection is not perfect, we have a reprojection error (which we aim to minimize)

The challenge arises from the nonlinear and high-dimensional nature of this problem, particularly in larger scenes with numerous parameters. Techniques like nonlinear least squares optimization are employed to tackle these challenges and obtain accurate 3D reconstructions. The mathematical formulation of this problem is given by the following equation :

$$\min_{C_j, X_i} \sum_{i=1}^{n} \sum_{j=1}^{m} \rho_{ij} \left\| P\left(C_j, X_i\right) - x_{ij} \right\|^2 \tag{2}$$

where X_i and C_j indicate a 3D point and a camera, respectively; $P\left(C_j, X_i\right)$ is the projection of point X_i on camera C_j; x_{ij} is an observed image point; $\|\bullet\|$ denotes the L2-norm; ρ_{ij} is an indicator function with $\rho_{ij} = 1$ if point X_i is visible in camera C_j; otherwise, $\rho_{ij} = 0$.

The entire processe of computing the SfM points is summarized in the fig3.

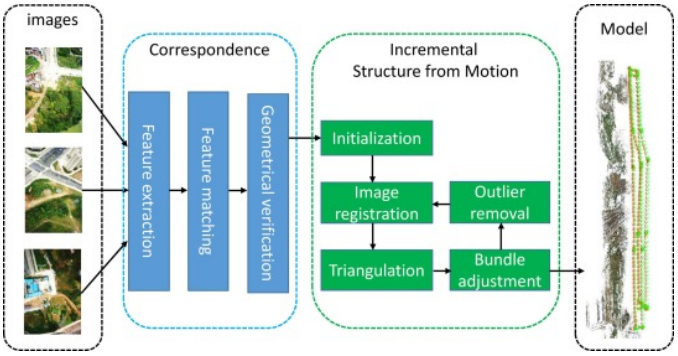


Fig. 3. Entire process of Structure from Motion [7]

B. MultipleView Stereo (MVS)

After applying Structure from Motion (SfM) to recover camera poses and sparse 3D points, Multi-View Stereo (MVS) plays a crucial role in achieving a complete and dense 3D model. MVS algorithms recover 3D geometric information much in the same way our visual system perceives depth by fusing two views. In the MVS setting, we may have many images that see the same point and could be potentially used for depth estimation. To recover a dense model, we estimate depths for every pixel in every image and then merge the resulting 3D points into a single model. The following figure shows how we pick the depth of a point we conflict occur between views of the same point : consistency among textures at these image projections is evaluated at each potential position. At the true depth (highlighted in green), the consistency score is at its maximum.

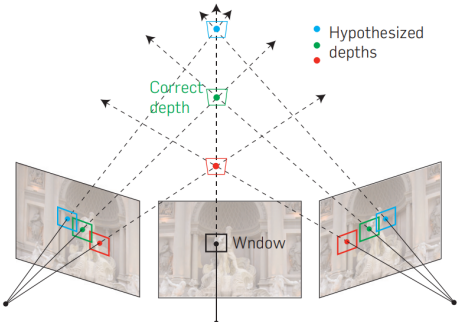


Fig. 4. Calculating the depth of point[4]

Many different MVS algorithms have been studied, each of them seem to be more adapted for a certain ttype of scene to reconstruct. MVS algorithms vary in how they represent geometry, commonly using voxels, level-sets, polygon meshes, or depth maps. The choice of scene representation impacts the subsequent reconstruction process. A robust MVS algorithm is proposed in [3], it involves a simple match, expand, and filter procedure 1) Matching pairs features across multiple images, generating a sparse set of patches for salient image regions. This process is repeated multiple times. 2) Expansion spreads initial matches to nearby pixels, creating a dense set of patches. 3) Filtering uses visibility and a mild form of regularization constraints to eliminate incorrect matches. This algorithm does not require a Sfm model, this is why the second step is not necessary when we have a Sfm model.

C. Capturing visual data for 3D reconstruction

In the realm of 3D reconstruction, the utilization of visual data plays a pivotal role, and one common approach involves the incorporation of videos. Videos offer a controlled environment for the Structure from Motion (SfM) and Multiple View Stereo (MVS) processes, providing a sequence of frames that contribute to a comprehensive understanding of the scene's geometry.

Indeed, the sequential nature of frames allows for the capture of the scene from various angles and perspectives. It also facilitates the correspondence step as we only have to compare a few successive frames.

The controlled capture of a video of the environment often pair images with additional information, such as GPS position data. By synchronizing the video frames with GPS coordinates, each frame becomes not only a visual snapshot but also a geospatially referenced data point. This integration of location information reduces significantly the complexity of the problem. Enhances the accuracy and contextual relevance of the reconstructed 3D model, especially when dealing with large-scale environments like cities.

Enhanced Constraints for SfM and MVS: The iterative refinement in Structure from Motion (SfM) benefits from the temporal coherence provided by consecutive frames in videos. Establishing correspondences between 2D projections becomes more robust, allowing for improved estimates of 3D points and camera parameters. Similarly, Multiple View Stereo (MVS) benefits from the dense sequence of frames, enhancing the completeness and precision of the 3D model.

The next figure (5) represents the dense reconstruction of an environment captured by a car equiped with a GPS sensor.

Fig. 5. Dense reconstruction of a city from a video taken by a car [2]

The subsequent section (IV) delves into the application of these 3D reconstruction techniques using less controlled data publicly available on the Internet.

IV. Reconstructing cities through the eyes of Internet

If we don't have access to a car that goes the city around equiped with camera or if we want to reconstruct a 3D model

of any city in the world we could want to use the images available for free in the internet. Indeed, for many touristic locations, we can find thousands of pictures of them online, this represents a large source of images taken from every angle. However working with large-scale datasets in the context of city-scale image matching poses significant constraints. The primary challenge arises from the sheer volume of data involved, with tens or hundreds of thousands of images in a collection. The task of finding correspondences across this extensive dataset is computationally demanding, and a naive approach of performing all pairwise image matches becomes impractical due to its quadratic time complexity. For instance, matching 100,000 images would require an unfeasible amount of time and computational resources. The inherent sparsity of the match graph, where most image pairs do not correspond, further complicates the situation. To address these constraints, [4] proposes a method to build a graph where the images viewing the same points would be linked. This method consist in the rapid creation of a sparse graph by comparing the photos globally with a method inspired of text document analysis and then testing iteratively if the ”close” vertex can be connected : if an image is connected to two other images, we verify if those two images can be connected. The next figure (6) shows the result of matching different views of a same point.

Fig. 6. Matching of views of the same point in images taken with different conditions [4]

Solving Equation of minimization (2) directly poses a challenging nonlinear optimization problem. Most of the time, Structure from Motion (SfM) methods applied to disordered photo collections adopt an incremental approach. They initiate with a modest reconstruction, and incorporate gradually a few images at each step, triangulating new points, and iteratively applying nonlinear least squares optimization, to minimize reprojection errors (this is commonly referred to as bundle adjustment). This cyclic process continues until it’s no longer possible to add more images. Nevertheless, for such large photo collections, implementing such an incremental approach simultaneously on all photos would be impossible. To deal with this problem, [5] propose to use a ”skeletal set” to perform the Sfm. Indeed, we can reduce substantially the number of images to process : many pictures are taken from the exact same point of view and do not add any additionnal information.

The MVS is also made impossible by the number of images to process at once. That is why we separate the computation of the MVS in different clusters that reconstruct each a small part of the scene.

Lastly an issue caused by the uncontrolled nature of the images is the determination of the color of the points of the model. Computing the color of a point in image-based 3D reconstruction poses several challenges, primarily stemming from variations in capture conditions, such as different cameras, times, and weather conditions. These discrepancies lead to color variations between images, making direct use of original images for texture mapping problematic and impacting the visual quality of 3D models. To address this challenge, the paper [6] proposes a solution based on sparse points obtained through Structure from Motion (SfM). By minimizing the color variance of these sparse points, the authors aim to create an efficient and effective color correction method.

The next figures (7, 8) shows the results of the 3D reconstruction Rome using images found on the Internet.

Fig. 7. 3D Reconstruction of the Colosseum using photos from internet *photo, Sfm points, MVS results from left to right*

Fig. 8. 3D Reconstruction of the St Peter Basilica using photos from internet

V. Conclusion

In conclusion, our survey has explored the fascinating realm of 3D reconstruction from multiple view images, using classical methods, such as Structure from Motion (SfM) and Multiple View Stereo (MVS).

It has explained integration of technologies like Scale-Invariant Feature Transform (SIFT) and Random Sample Consensus (RANSAC) to establish the correspondences between images, laying the groundwork for accurate 3D reconstructions.

In the end, reconstructing cities through the eyes of the internet presents exciting opportunities but also substantial challenges. The sheer volume of data requires innovative approaches, such as graph-based methods and skeletal sets, to efficiently find correspondences across extensive datasets. Overcoming computational constraints and addressing color variations in uncontrolled images are crucial steps toward achieving high-fidelity 3D reconstructions.

References

[1] Lowe, D. G. (1999). ”Object Recognition from Local Scale-Invariant Features.” In: Proceedings of the Seventh IEEE International Conference on Computer Vision, Kerkyra, Greece, 1999, pp. 1150-1157 vol.2. DOI: 10.1109/ICCV.1999.790410. Keywords: Object recognition, Electrical capacitance tomography, Image recognition, Lighting, Neurons, Computer science, Reactive power, Filters, Programmable logic arrays, Layout.

[2] Akbarzadeh, A., et al. "Towards Urban 3D Reconstruction from Video." In: Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06), Chapel Hill, NC, USA, 2006, pp. 1-8. DOI: 10.1109/3DPVT.2006.141.

[3] Furukawa, Y., Ponce, J. "Accurate, Dense, and Robust Multiview Stereopsis." In: IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 32, No. 8, August 2010.

[4] Agarwal, S., Furukawa, Y., Snavely, N., Simon, I., Curless, B., Seitz, S. M., Szeliski, R. "Building Rome in a day." Communications of the ACM, 54(10), October 2011, 105–112. DOI: 10.1145/2001269.2001293.

[5] Snavely, N., Seitz, S. M., Szeliski, R. "Photo tourism: exploring photo collections in 3D." In: ACM SIGGRAPH 2006 Papers (SIGGRAPH '06), New York, NY, USA, 2006, 835–846. DOI: 10.1145/1179352.1141964.

[6] Yang, J., Liu, L., Xu, J., et al. "Efficient global color correction for large-scale multiple-view images in three-dimensional reconstruction." ISPRS Journal of Photogrammetry and Remote Sensing, 2021, Vol. 173, p. 209-220.

[7] Jiang, S., Jiang, C., Jiang, W. (2020). "Efficient Structure from Motion for Large-Scale UAV Images: A Review and a Comparison of SfM Tools." ISPRS Journal of Photogrammetry and Remote Sensing, 167, 230-251.

Autonomous Sailboat Navigation Using Potential Fields

Johan BERRIER-GONZALEZ, *Student, ENSTA Bretagne*

***Abstract*—The artificial potential field method is a widely used algorithm for trajectory planning of autonomous ground robots.**

In this article, this method is applied to local path planning of an autonomous sailing robot. The environment and constraints specific to sailboat navigation (restricted areas upwind and downwind) are represented by local potentials built around the boat's location and periodically updated to account for changes in wind direction and obstacle positions.

The method we apply in this article is a dynamic potential field that adapts according to the size of the obstacle, recalculating the course vector needed to avoid the obstacle and reach the desired objective at each point on the map.

***Index Terms*—Unmanned Surface Vehicles, Path Planning, Potential Field Methods, Simulation, Review**

I. Introduction

Unmanned Surface Vehicles (USVs) have garnered significant attention in recent years due to their broad applications in marine research, environmental monitoring, surveillance, and maritime transport. These autonomous or remotely operated vessels offer several advantages over traditional manned vessels, including cost-effectiveness, safety, and the ability to access remote or hazardous environments.

One of the primary challenges associated with operating USVs is efficiently planning routes to navigate complex marine environments while avoiding obstacles and reaching predefined destinations. Traditional trajectory planning approaches often rely on predefined maps or global trajectory planning algorithms, which may not be suitable for dynamic environments or real-time applications.

Recent advancements in trajectory planning techniques have led to the development of new algorithms specifically tailored to USV navigation. These algorithms often leverage concepts from artificial intelligence, optimization, and control theory to generate optimal or near-optimal trajectories in dynamic marine environments.

This navigation method, based on artificial potential fields, is designed to react in real-time to changes in the environment (such as wind direction, obstacles like other boats, trees, etc.) while also adapting to the specific kinematic constraints of the yacht (represented by its speed polar). The algorithm calculates a feasible course at each instant.

The article is organised as follows:

- Course Controller
- Attractive and Repulsive Potential Fields
- Attractive Line
- Obstacle Avoidance
- Simulation and Results

II. Preliminary Knowledge

Over the past decade, numerous methods have been proposed to address the navigation and obstacle avoidance challenges of autonomous sailing robots. Ray-tracing techniques have been proposed in [3], [4], the utilization of a state machine in [5], [6], modification of the A* algorithm in [7], application of fuzzy logic in [8], and adoption of Voronoi diagrams in [9]. Additionally, the implementation of Velocity Made Good (VMG), a classical navigation method, was discussed in [10].

In these methods, the goal is to navigate while considering obstacles such as other boats, trees, etc., and adapt to the specific kinematic constraints of the sailboat, represented by its velocity polar. At each moment, the algorithm computes a feasible heading, ensuring the vehicle remains within a band surrounding the direct route between consecutive waypoints determined by a human operator.

Potential field methods, pioneered by Khatib [11] and Krogh [12], are commonly employed for motion planning of mobile robots (e.g., see [13]-[14] among others) and were first adapted for sailing robots in [15].

It is assumed that the mission planning task (global planning) has been completed, and a list of waypoints *Wp* has been selected either by a human mission planner or commercially available software. Additionally, a map containing known obstacle positions (e.g., islets, fixed moorings, etc.) is provided to the local route planner.

Utilizing this list of waypoints, the local path planner computes in real-time a feasible heading and sail angle to reach the desired waypoint while avoiding obstacles.

III. Cap controller

We have a controller which takes as input the objective to be reached (the course to follow) as well as the position, course and wind angle provided by the sailboat.

At the output, the system provides the following values to control the actuators:

- δ_{smax} (flap angle)
- δ_{r} (rudder angle)

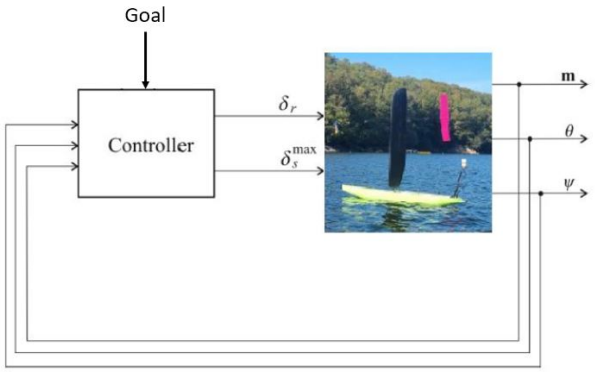


Fig. 1: Control law

Algorithm 1 Cap Control

Input:
m (robot position)
w (course to follow)
ψ (wind angle)
ψ_{ap} (apparent wind angle)
θ (current heading angle of the robot)
w_1 (x component of heading vector to follow)
w_2 (y component of the heading vector to follow)

Output:
u (control command)
Calculating the desired heading angle
$\theta_{bar} = \arctan 2(w_2, w_1)$

Rudder control
$\delta_r = 0.3 \times \text{sawtooth}(\theta - \theta_{bar})$

Sail control
$\delta_s bar = -\text{sign}(\sin(\psi - \theta)) \times (\pi/4) \times (\cos(\psi - \theta) + 1)$
$\delta_s = \text{sawtooth}(\delta_s - \delta_s bar)$
$u_2 = \arctan\left(\frac{\cos(\delta_s - \psi_{ap}) \times \delta_s}{1 - \sin(\delta_s - \psi_{ap}) \times \delta_s}\right)$

Total control
$u = [\delta_r, u_2]^T$

This is the control algorithm we will use throughout this article for the various potential fields.

IV. Potential fields

A. Attractive and repulsive potential fields

In artificial potential field methods, the movement of the robot (represented as a particle) is guided by a field consisting of two main components: an attractive potential that pulls the robot towards the goal, and a repulsive potential that pushes it away from obstacles. The primary drawback of potential field techniques is their susceptibility to local minima. However, for ocean surveillance missions, marine environments are typically sparse, and local minima are not a significant issue.

In the potential field approach, we establish an attractive field directed towards the target. This potential field is defined across the entire free space, and at each time step, we compute the potential field at the robot's position and then determine the force exerted by this field.

Algorithm 2 Attractive potential

$d = \sqrt{(X - x_g)^2 + (Y - y_g)^2}$
$\theta = \arctan 2(Y - y_g, X - x_g)$.
if $d < 0$ **then**
  delx $= 0$
  dely $= 0$
else
  delx $= -\alpha \times d \times \cos(\theta)$
  dely $= -\alpha \times d \times \sin(\theta)$
end if
Where:

- d is the distance between the current point (X, Y) and the goal (x_g, y_g) .
- α attractivity coefficients

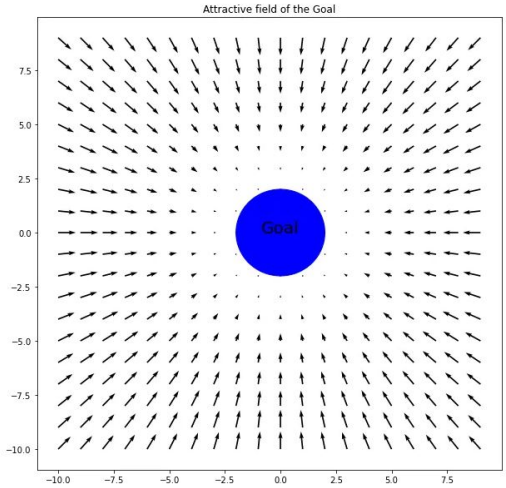


Fig. 2: Fields of attractive potential

We can also define another behaviour that enables the robot to avoid obstacles. We ensure that each obstacle generates a repulsive field around it. If the robot approaches the obstacle, a repulsive force will act on it, moving it away from the obstacle.

We use a method in which we introduce two parameters *r* the safety distance between the obstacle and the boat and *s* the range coefficient of the repeller. We use a method, in which we introduce two parameters : *r* the safety distance between the obstacle and the boat and *s* the range coefficient of the repeller.

Algorithm 3 Repulsive potential

$d = \sqrt{(X - x_g)^2 + (Y - y_g)^2}$
$\theta = \arctan 2(Y - y_g, X - x_g)$.
if $d < 0$ **then**
  delx $= sign(\cos(\theta))$
  dely $= sign(\sin(\theta))$
else if $d > r + s$ **then**
  delx $= 0$
  dely $= 0$
else
  delx $= \beta \times (s + r - d) \times \cos(\theta)$
  dely $= \beta \times (s + r - d) \times \sin(\theta)$
end if
Where:

- d is the distance between the current point (X, Y) and the goal (x_g, y_g)
- r is the safety distance
- s is the range of the repulsor
- β repulsivity coefficients

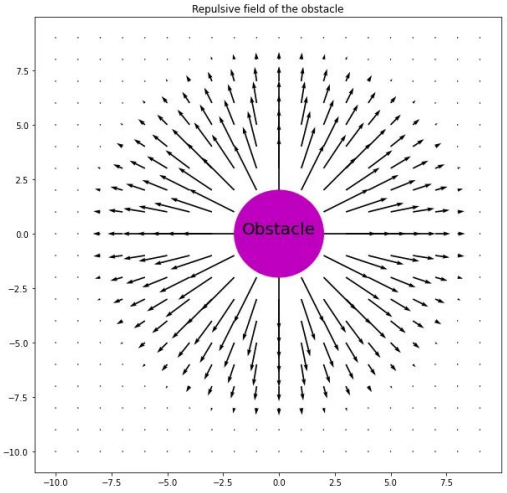


Fig. 3: Fields of repulsive potential

B. Following Lines

The article [1] highlights the problem of line tracking when the wind remains constant and the line to be tracked is upwind. It explores different strategies for solving this problem.

Here are the results he obtained:

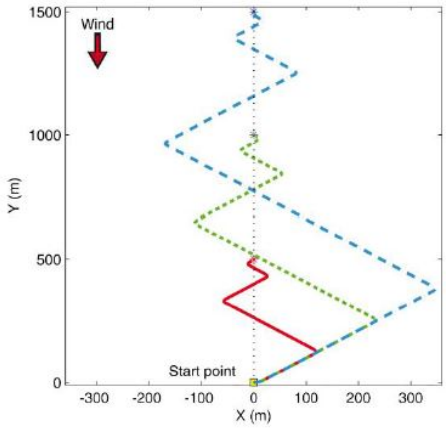


Fig. 4: Simulation results for upwind navigation, for a waypoint at a distance of distance of 500 m, 1 000 m and 1 500 m

In the rest of the article he proposes a particularly interesting strategy, which consists of defining the objective as a virtual target moving along the attractive line instead of reaching a static point. This allows the target to be dynamically adapted to the desired trajectory.

To constrain the vehicle to stay within a band-limited region around the direct route between two consecutive waypoints, the previous method is modified using a line-of-sight (LOS) guidance approach [1]. In this case, the attractive potential is no longer associated with the current waypoint but with a LOS point, which improves the stability and accuracy of line-of-sight tracking.

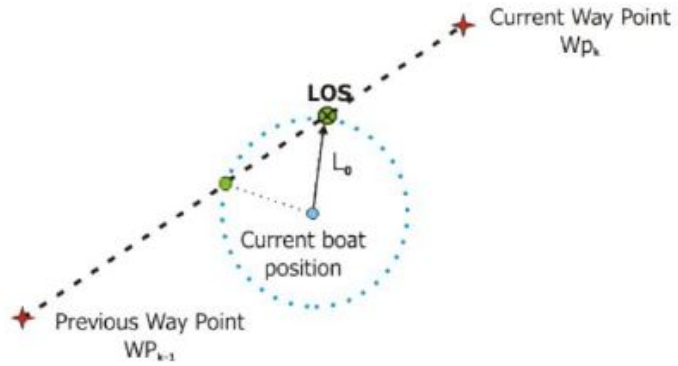


Fig. 5: Line of sight (LOS) point definition)

Here are the results he obtained :

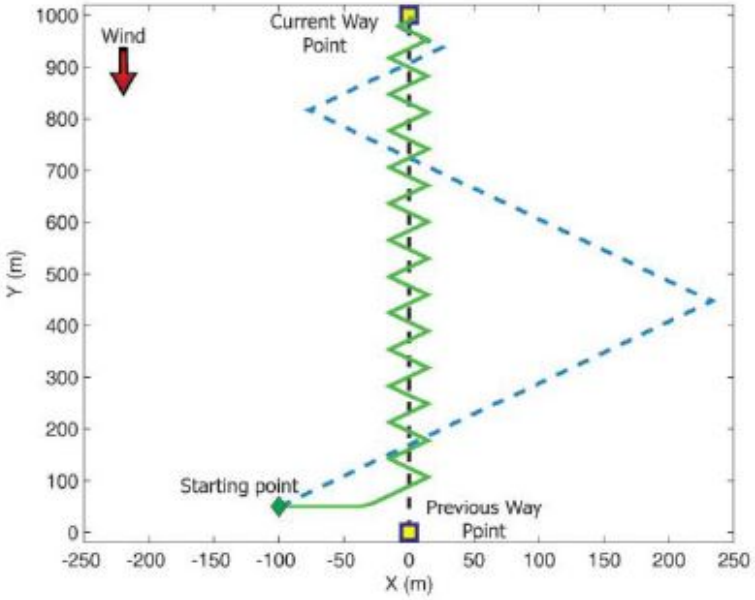


Fig. 6: Attractive line tracking

By using this strategy with my method of integrating potential fields, I obtain similar results :

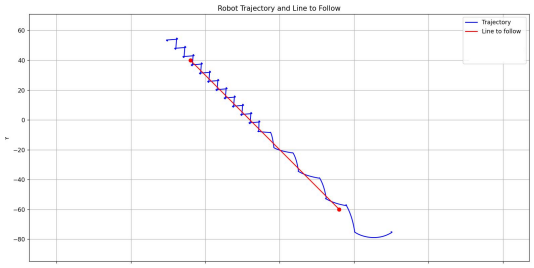


Fig. 7: Results Attractive line tracking

C. Avoidable obstacle

[2] uses a potential field that adapts to the shape of the object, allowing it to adapt its trajectory to the shape of the object.

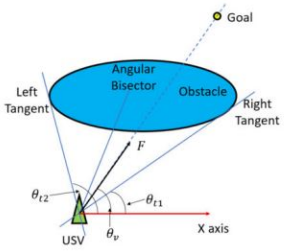


Fig. 8: Diagram of adaptive repulsive potential fields

The disadvantage of this method is that it is mathematically very complex and expensive to calculate.

We use a simpler method, presented in the section on attractive and repulsive potential fields, in which we introduce two parameters : *r* the safety distance between the obstacle and the boat, and *s* the range coefficient of the repeller.

Algorithm 4 Potential Field Calculation

Initialize s, r

Initialize $\Delta x = 0$, $\Delta y = 0$

$d_{\text{goal}} = \sqrt{(x_{\text{goal}} - X)^2 + (y_{\text{goal}} - Y)^2}$

$d_{\text{obstacle}} = \sqrt{(x_{\text{obstacle}} - X)^2 + (y_{\text{obstacle}} - Y)^2}$

$\theta_{\text{goal}} = \arctan 2(y_{\text{goal}} - Y, x_{\text{goal}} - X)$

$\theta_{\text{obstacle}} = \arctan 2(y_{\text{obstacle}} - Y, x_{\text{obstacle}} - X)$

$$\Delta x_{\text{goal}} = \begin{cases} 0 & \text{if } d_{\text{goal}} \leq 0 \\ -\alpha \times s \times \cos(\theta_{\text{goal}}) & \text{if } d_{\text{goal}} > 0 \end{cases}$$

$$\Delta y_{\text{goal}} = \begin{cases} 0 & \text{if } d_{\text{goal}} < 0 \\ -\alpha \times s \times \sin(\theta_{\text{goal}}) & \text{if } d_{\text{goal}} > 0 \end{cases}$$

$$\Delta x_{\text{obstacle}} = \begin{cases} \text{sign}(\cos(\theta_{\text{obstacle}})) & \text{if } d_{\text{obstacle}} < r \\ 0 & \text{if } d_{\text{obstacle}} > r + s \\ \beta(s + r - d_{\text{obstacle}})\cos(\theta_{\text{obstacle}}) & \text{otherwise} \end{cases}$$

$$\Delta y_{\text{obstacle}} = \begin{cases} \text{sign}(\sin(\theta_{\text{obstacle}})) & \text{if } d_{\text{obstacle}} < r \\ 0 & \text{if } d_{\text{obstacle}} > r + s \\ \beta(s + r - d_{\text{obstacle}})\sin(\theta_{\text{obstacle}}) & \text{otherwise} \end{cases}$$

$\Delta x+ = \Delta x_{\text{goal}} + \Delta x_{\text{obstacle}}$

$\Delta y+ = \Delta y_{\text{goal}} + \Delta y_{\text{obstacle}}$

The results obtained :

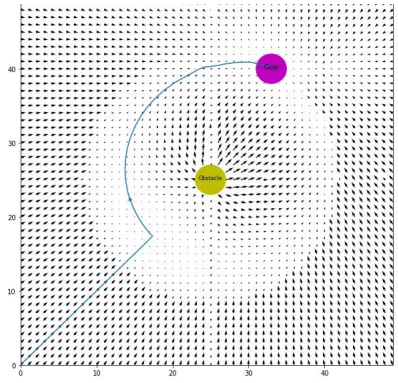


Fig. 9: Adaptive obstacle avoidance

[2] gives the following result:

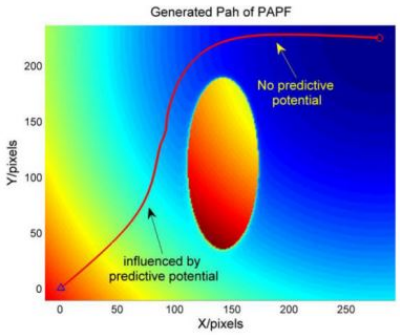


Fig. 10: Results obstacle avoidance

If we compare the two results we can see that [2] adapts better to the shape of the object but the result I get also works with a less expensive calculation time.

V. Results

Here are the simulation results obtained when implementing my potential field algorithm.

I have created a python simulator in which I simulate the environment, namely :

- the true wind and the apparent wind represented by the red and green vectors
- the obstacle represented by a red dot placed in the middle of the line
- the attractive line
- the corresponding potential field and therefore all possible set points

I also simulates the sailboat following the line thanks to its course controller.

You can see that the potential field adapts according to the obstacle.

I also plotted the trajectories followed by the sailboat by varying the parameters *r* (safety distance) and *s* (range of the

repeller).

We can see that the parameters s and r influence the distance at which the sailboat avoids the obstacle, as illustrated on the blue trajectory.

These parameters should therefore be chosen according to the situation and the type of obstacle to be avoided.

The trajectory in black which corresponds to a r =20 and a s =40 simulates a very imposing obstacle, which allows us to verify that even in such a situation, the sailboat always tends to recover its attractive line.

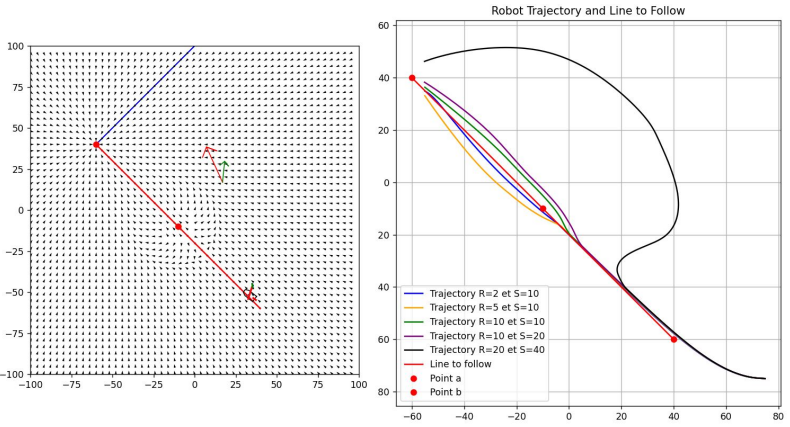


Fig. 11: Simulation result

VI. Conclusion

A new method for line-of-sight tracking of an autonomous sailboat has been presented in this article, combining line-of-sight guidance and the potential terrain method. This method has been validated using a python simulation and demonstrates good performance. In addition, a new approach for dynamically defining a potential field by defining the size of the repulsive potential has been presented. Although this algorithm is simple to implement, it does have some limitations in terms of the choice of r and s and the repulsivity coefficients.

References

[1] [1] Frederic Plumet, Hadi Saoud, Minh-Duc Hua "Line following for an autonomous sailboat using potential fields method," June 2013

[2] [2] Jia Song1, "Path planning for unmanned surface vehicle based on predictive artificial potential field", Febraury 2020

[3] [3] C. Sauze and M. Neal, "A raycast approach to collision avoidance in sailing robots," in International Robotic Sailing Conference, 2010, pp. 25–32

[4] [4] R. Stelzer, K. Jafarmadar, H. Hassler, R. Charwot et al., "A reactive approach to obstacle avoidance in autonomous sailing." in International Robotic Sailing Conference, 2010, pp. 33–39.

[5] [5] Y. Briere, "IBoat: an unmanned sailing robot for the Microtransat challenge and ocean monitoring," in Conf. TAROS (Towards Autonomous Robotic Systems), 2007, pp. 145–162.

[6] [6] G. Elkaim and R. Kelbley, "Station keeping and segmented trajectory control of a wind-propelled autonomous catamaran," in 45th IEEE Conference on Decision and Control, 2006, pp. 2424–2429.

[7] [7] H. Erckens, G.-A. Busser, C. Pradalier, and R. Siegwart, "Avalon: Navigation strategy and trajectory following controller for an autonomous sailing vessel," IEEE Robotics Automation Magazine, vol. 17, no. 1, pp. 45 –54, 2010.

[8] [8] R. Stelzer, T. Proll, and R. John, "Fuzzy logic control system for autonomous sailboats," in IEEE Int. Conf. on Fuzzy Systems, 2007, pp. 1–6.

[9] [9] K. Xiao, J. Sliwka, and L. Jaulin, "A wind-independent control strategy for autonomous sailboats based on Voronoi diagram," in 14th Int. Conference on Climbing and Walking Robots (Clawar2011), 2011, pp. 110–124.

[10] [10] R. Stelzer and T. Proll, "Autonomous sailboat navigation for short course racing," Robotics and Autonomous Systems, vol. 56, no. 7, pp. 604–614, 2008.

[11] [11] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," International Journal of Robotics Research, vol. 5, pp. 90–98, 1986.

[12] [12] B. Krogh and C. Thorpe, "Integrated path planning and dynamic steering control for autonomous vehicles," IEEE Int. Conf. on Robotics and Automation, pp. 1664–1669, 1986

[13] [13] J. Barraquand, B. Langlois, and J. Latombe, "Numerical potential field techniques for robot path planning," IEEE Transactions on Systems, Man and Cybernetics, vol. 22, pp. 224–241, 1992.

[14] [14] S. Shimoda, Y. Kuroda, and K. Iagnemma, "Potential field navigation of high speed unmanned ground vehicleson uneven terrain," IEEE Int. Conf. on Robotics and Automation, pp. 2828 – 2833, 2005.

[15] [15] C. Petres, M. Romero Ramirez, and F. Plumet, "Modeling and reactive navigation of an autonomous sailboat," in IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011, pp. 3571–3576.

Set-membership state estimation using mechanically scanning sonar systems

Gabriel Betton
ENSTA Bretagne, Lab-STICC, UMR CNRS 6285, Brest, France
Brest, France

***Abstract*—This paper deals with the localization problem of a mobile robot in a known environment using low-cost sensors. The initial position is unknown and measurements are uncertain. The solution suggested here is the use of an interval analysis, a non pessimistic set-membership approach to deal with uncertainties.**

***Index Terms*—state estimation, interval analysis, sonar**

I. Introduction

The use of mechanically scanning sonar systems (MSIS) is particularly suitable for use on small, low-cost ROVs for navigation and obstacle avoidance. This article proposes a solution for locating a mobile robot using the data received by this type of sensor. Given the environment and the physical properties of the sensor, measurement errors and uncertainty on the state will be present. To take these into account, this article presents the use of an interval analysis [5].

We assume that the map of the environment is available and is made of static, indistinguishable segments. The initial position of the robot is unknown. For this first approach the problem will be kept in a plan.

Mechanically scanning sonar systems (MSIS) continuously scans its environment, one full 360° scan takes several seconds during which the robot continues to move. Different approaches exist to use this data to estimate the localization of a robot, one of the techniques widely used is the Iterative Closest Point (ICP) [1] or its probabilistic extension (pIC) [2] [4] to take into account uncertainties. Both aim at computing the given displacement between sets of acquired 3D point clouds to estimate the robot displacement.

Our approach is to process this localization problem using a constraint network, each measurement is taken independently to contract the robot trajectory at given times. This method should save computational power while guaranteeing a certain trajectory within a set without requiring linearization or Gaussian distributions.

This paper will develops the constraint network used for the localization and presents a simulation for demonstration.

II. Formalism

The map is known beforehand, let it be defines by $\mathbb{M}$, a set of vectors :

$$\mathbb{M} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$$

where $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ are vectors describing the segments of the map.

The localization problem corresponds to the following state estimation of the robot:

$$\begin{cases} \dot{x}(t) = f(x(t), u(t)) & \text{(evolution equation)} \\ y^i = g(x(t^i)) & \text{(observation equation)} \end{cases}$$

where x is the unknown state vector (, u is the input measurement vector, and y^i is an output measurement vector made at time t^i. As x and u continuously evolve with time, we define them as trajectories denoted by $x(\cdot)$ and $u(\cdot)$, contrary to y^i that represents discrete data.

For our interval analysis we assume that all measurements errors can be bounded so we can note $y^i \in [y^i]$.

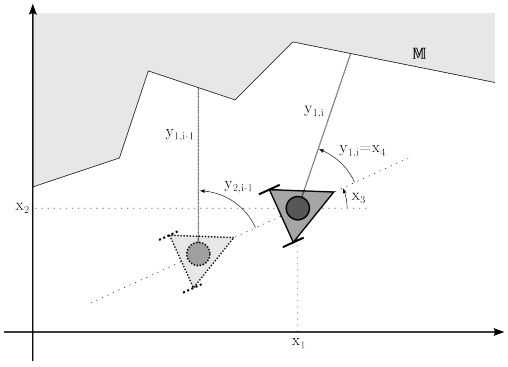


Fig. 1. The map $\mathbb{M}$ is perceived by the robot at a time t^i and t_{i-1}

III. Constraint Network

Evolution equation. Let's consider a robot moving on a plane, for our demonstration we use a unicycle model with the sensor angle as the fourth state :

$$\dot{x}(t) = f(x(t), u(t)) = \begin{pmatrix} v_r(t) \cdot \cos(x_3(t)) \\ v_r(t) \cdot \sin(x_3(t)) \\ \omega_r(t) \\ \omega_s(t) \end{pmatrix}$$

where $u(t) = \begin{pmatrix} v_r(t) & \omega_r(t) & \omega_s(t) \end{pmatrix}$ with $v_r(t)$ the robot speed, $\omega_r(t)$ the robot rotation rate and $\omega_s(t)$ the sonar rotation rate.

Observation. A MSIS acquires a set of range-bearing measurements y^i at discrete times t^i. The bearing is the current angle of the sonar and the range is the distance from the robot to the segment belonging to the map $\mathbb{M}$.

$$y^i = g(x(t^i)) = \begin{pmatrix} dist(x(t^i), \mathbb{M}) \\ x_4(t^i) \end{pmatrix}$$

where $dist(x(t^i), \mathbb{M})$ returns the distance between the robot and the closest wall at the angle $x_4(t^i)$, the algorithm to compute the observation vector in a plane environment is described as follows :

Algorithm 1 Observation

Input: $\mathbb{M}, t^i, x$
Output: $y(t^i)$
Initialisation :
compute sonar vector
min_distance = ∞
Loop
for m in $\mathbb{M}$ **do**
  if sonar vector intersects m **then**
    if intersection point in m **then**
      distance = dist($x(t^i)$, intersection_point)
      if distance ¡ min_distance **then**
        min_distance = distance
      end if
    end if
  end if
end for
return (min_distance , $x_4(t^i)$)

Contractors A set of contractors is introduced to estimate the localization of the robot, these constraints will then be used for the solver.

$$\begin{cases} (i) & c^i = x(t^i) \\ (ii) & a^i = y_1^i \begin{pmatrix} \cos(y_2^i) \\ \sin(y_2^i) \end{pmatrix} \\ (iii) & b^i \in \mathbb{M} \\ (iv) & c^i = b^i - a^i \end{cases}$$

Intermediate variables are introduced to decompose the equations and reveal elementary constraints [5], $a^i \in \mathbb{R}^2$ is the displacement vector corresponding to the range and bearing measurements (y_1^i, y_2^i), $b^i \in \mathbb{R}^2$ is the intersection point of the sonar vector with the first segment of $\mathbb{M}$, $c^i \in \mathbb{R}^2$ is the position $(x_1(t^i), x_2(t^i))$ of the robot at the time t^i.

We use contractors whose solutions have already been defined in software libraries :

- *(i)* evaluates the trajectory $x(\cdot)$ at a given time t^i and create a constraint with the vector a^i. A dedicated contractor C_{eval} has been provided in [6]. This is the main contractor that allows to use the discrete measurements of a MSIS to estimate a trajectory.
- *(ii)* links a range and bearing measurement to a a vector of distance (Polar coordinates to Cartesian). The contractor C_{polar} introduced in [3] is used.
- *(iii)* links a point to a set of segments, the contractor used is an union of all segment contractor $C_{segment}$ of the map $\mathbb{M}$. Let it be defined by $C_{map} = \bigcup_{m \in \mathbb{M}} C_{segment}(m)$
- *(iv)* we use a simple difference contractor C_- to link a^i, b^i and c^i.

IV. Application

All domains are initialized as set of all real values except for measurements and their bounded uncertainties $y^i \in [y^i]$. All contractors defined are used in the contractor network. For each measurement i :

$$\begin{cases} (i) & C_{eval}([t^i], [c^i], [x](\cdot)) \\ (ii) & C_{polar}([a_1^i], [a_2^i], [y_1^i], [y_2^i]) \\ (iii) & C_{map}([b^i]) \\ (iv) & C_-([c^i], [b^i], [a^i]) \end{cases}$$

The goal of the following demonstration is to estimate a set of possible trajectories of the vehicle, in a real case, an extended Kalman filter (EKF) can be added to more accurately estimate the position of the robot using previously estimated initial position.

The solver is illustrated on a simulated example where a robot move along among a map made of 4 segments, the robot is equipped with a MSIS rotating at ω_s=2 rad/s and measuring the distance to the facing wall each 200 ms. The robot follows a Lissajous trajectory :

$$x(t) = \begin{pmatrix} \cos(t) \\ 2 \cdot \sin(2 \cdot t) \end{pmatrix}$$

We use the CODAC library made by S. Rohou to solve the contractor network along with VIBEs for visualization. When running the simulation the contraction is made in 0.06s :

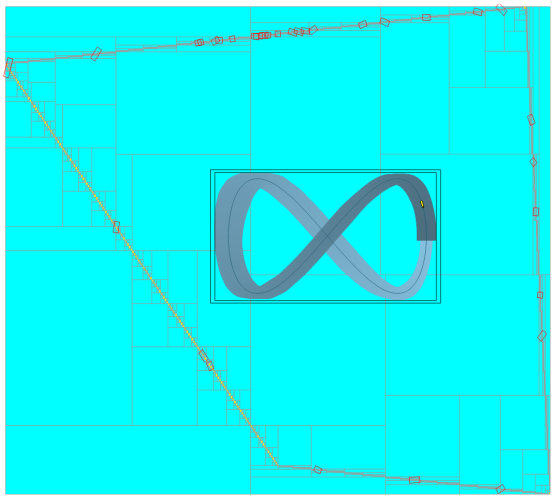


Fig. 2. Map of the simulated mission, the actual trajectory is the black line, measurements are shown on red, a SIVIA paving is used to visualize the map

The Lissajous trajectory demonstrates an undesired behavior. If the robot is turning in the opposite direction of the MSIS at the same rotation rate, the sonar always faces the same direction and returns the same measurement not contracting the trajectory. A solution could be to drive the MSIS at a rotation rate $\omega_s = K + \omega_r$ to ensure a fixed rotation rate in the world frame. Unfortunately common commercial MSIS do not have this type of control.

V. Conclusion

This method using interval analysis is quick and reliable to solve the localization problem of underwater robot using MSIS in presence of uncertainties and unknown initial localization. A reservation is made concerning fast-turning robots that could make the MSIS unusable.

Considering the computation time, this algorithm can also be used for online localization in a known environment.

Libraries

The CODAC is a C++/Python library providing tools to guarantee computations over intervals of reals, sets and trajectories. The official web page is www.codac.io .

Acknowledgments

We thank S. Rohou for its support and precious advices during this interval analysis experimentation. We also thank him for the developement of the CODAC library.

References

[1] P.J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[2] Yohan Breux, André Mas, and Lionel Lapierre. On-manifold probabilistic iterative closest point: Application to underwater karst exploration. *The International Journal of Robotics Research*, 41(9-10):875–902, 2022.

[3] B. Desrochers and L. Jaulin. A minimal contractor for the polar equation: Application to robot localization. *Engineering Applications of Artificial Intelligence*, 2016.

[4] Luis Montesano, Javier Minguez, and Luis Montano. Probabilistic scan matching for motion estimation in unstructured environments. pages 3499 – 3504, 09 2005.

[5] Simon Rohou, Benoît Desrochers, and Luc Jaulin. Set-membership state estimation by solving data association. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4393–4399, 2020.

[6] Simon Rohou, Lyudmila Mihaylova, Luc Jaulin, Fabrice Bars, and Sandor Veres. Reliable non-linear state estimation involving time uncertainties. *Automatica*, 93, 01 2018.

Stalactite detection using Hough Transform

S. Bourzoufi[1]

Lab STICC, ENSTA Bretagne
2 rue François Verny, 29200 Brest, France

E–mail: *samy.bourzoufi@ensta-bretagne.org*

SUMMARY: The purpose of this paper is to use line detection technique with Hough transform in order to detect stalactites for a robot explorating a cave. Caves being dark, it is nearly impossible to obtain good results by applying directly the line detection. We suggest an image processing method in order to make stalactites more detectable and see its effect on the Hough Transform

Key words. Hough transform – Line detection – vision – image processing

1. INTRODUCTION

Among the shape detection techniques in vision, the Hough transform is a celebrity. First introduced by Paul Hough in 1959 [1], to detect lines in a hydrogen bubble chamber. However, the affine parametrization of lines poses problems, Duda and Hart [2] propose a polar parametrization called the "rho-theta" parametrization solving the said problems.

The Hough transform is a mathematical object which, in general, transforms a point on the plane into a curve in a parameter space. A curve in the chosen parameter space therefore represents the set of straight lines of the initial plane passing through the chosen point. Thus, if we look at the first formalism chosen by Hough, the image of a point is a line in the parameter space, and the inverse transform of a point in the parameter space is a line in the original plan. Thus, the idea behind the line detection algorithm is to apply the Hough transform to all the singular points of an image (found with Canny for example) then to calculate the density of the graphs of the associated lines (with an accumulation matrix): the important slopes intersections are then found in the accumulator and give the parameters of the line of the plane. No matter how the lines of the plane are parameterized, the method remains the same, it is called the "slope intersect" method. The rho-theta parameterization gives, in the parameter space, sinusoids whose intersections give a straight line of the plane. The issue in our problem is the cloud of caracteristic points we find in an image : the image being noisy, a lot of non-coherent lines are found. Using image processing and Hough transform properties, we are able to remove the incoherent lines and detect only (or almost) stalactites.

The idea is mainly to reduce the number of characteristic points of an image to improve the separability of the lines in the accumulator, but also to reduce the execution time of the process, this technique being intended to be used in real time by a autonomous robot. The group of line found on the image can be used by the robot to locate itself in the cave.

2. Hough transform on a raw image

In this paper we will work on the following cave image. There are a large number of more or less fine stalactites, with the rest of the cave being darker in the background. This image was chosen because it was very irregular in terms of shape and brightness. The image is 1170x780 in size and although it is in color, we will convert it to black and white first.

The easiest way to apply line detection is to use

Fig. 1: Raw image of a cave with stalactites.

the OpenCV2 library on python. The first thing to do is to convert the image from RGB to black and white level. Then we need to detect the edges from our black and white image: Canny edge detector is a pretty good way to do so. Here is what the image we get applying this method.

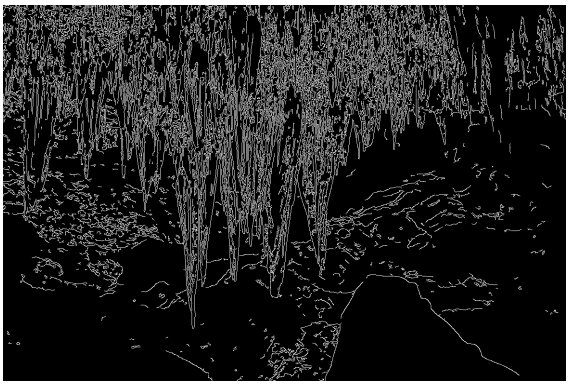


Fig. 2: Edges detection applied to our image using Canny.

As we can see, the edge detector gives us our image with edges in white. Of course, edges are a group of pixels with coordinate, which we want to apply Hough Transform to. Let's have a quick reminder of what the Hough transform is. If we consider the 'rho-theta' parametrization described by Duda and Hart [2], here is how the transformation can be defined:

$$H : \mathbb{R}^2 \to C([0, 2\pi])$$
$$(x, y) \mapsto \rho_{(x,y)}(\theta) = x\cos(\theta) + y\sin(\theta)$$

Indeed, a point from the 2D plane has for image a continuous function on the segment $[0, 2\pi]$ describing the parameters of all the lines intersecting that point. The parameters of a lines are the radius and its angle. As a consequence, if we apply Hough transform to all of our n edges points we obtain n ρ-functions. We then calculate the accumulator associated with these slopes and find the most rated lines parameters. By the way it is possible to visualize the parameters slopes with the accumulator, we'll see some example later. Here is the final picture with the line found with the algorithm :

Fig. 3: Lines found with our Hough transform process.

Obviously the result is not great, and still, this is not the worst result as a threshold has been applied to only show the most rated lines on the accumulator. This is however an interesting result as even the most obvious line a human would draw by himself on the original image are not found with Hough transform. The problem is that we select the wrong parameters in the accumulator and there are two possible reasons for that. First, we don't select the good maximum in our accumulator, which is not really likely to happen. The other reason would be that the accumulator is incremented in a way that its hard to separate certain slopes intersection, maybe because there are too much slopes. This makes sense as a large part of the edges points found with the Canny algorithm are not part of a noticeable line. The solution is then to process our image in a way that we only obtain the edges from the stalactites. Duda and Hart say in their paper [2] that the number of intersections in the parameter space increase in quadratic growth with the number of slopes, so reducing the number of edges points should be efficient.

3. Image processing for vertical edges detection

In order to reduce the number of edges points, we need to know what makes an edge point irrelevant in our context. First, if an image has very noisy light variation, applying a gradient to find edges will produce high spikes which we want to avoid. To solve this problem, we need to flatten the image in term of luminosity level. For instance, we can reduce the

number of intensity level and modify our pixels' intensity values to them using threshold. Instead of having 256 intensity level, we only take 3 levels for example (0, 125 or 255). As differences in depth produce differences in luminosity, edges will be detected more easily. However if a spike in luminosity happens due to a drop of water for example, we need to remove it as well. One way to do that is to "dilute" the spike of luminosity by applying a gaussian blur. Repeating this process of multithresholding and blurring will produce a fairly flat image in term of luminosity. If we take the hypothesis that the vertical axis of our robot's camera is colinear with the vertical lines of the stalactites (our robot is not necessarly navigating on a flat space so there is no really other way to formulate this hypothesis) we can pay attention only to vertical edges. By taking account of this hypothesis, we can apply a horizontal gradient in order to find vertical edges.

To find the horizontal gradient of an image we can use a horizontal gradient mask and convolute our image with it.

$$I_y = \partial y * I$$

With I_y the horizontal gradient of our image, I the original image and ∂y the mask of horizontal gradient :

$$\partial y = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

A final threshold is applied to the image in order to get a binary image (which is necessary to apply Hough transform). Here is what we obtain after this preprocess :

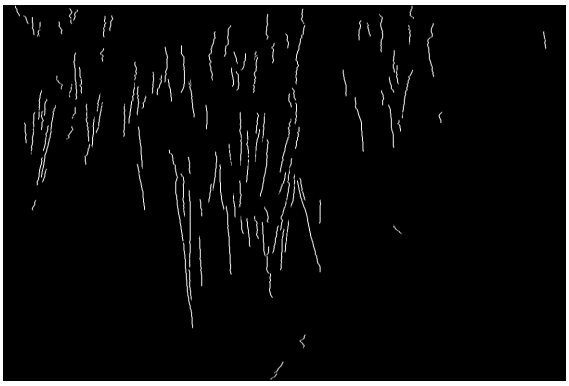


Fig. 4: Our image after the preprocess is applied.

As we were expecting, a lot of edges point have been remove or ignored compared to the result obtained with a Canny edge detector. The image processing can however produce a loss of precious information, for instance big stalactite on the gradient image only have one edges represented : this is due to luminosity on a global scale as both side of a stalactite might no be exposed to light the same way. Moreover, we still notice some edges points that are not really a part of noticeable lines, and this might appear to be a problem, but not really actually because the intersection of their slopes in the parameter space will be ignored thanks to the threshold. However, it is easy to get rid of that remaining noise : we compute all the group of neighbor pixels, if the group is too small, we set the value of its pixels to zero. Again, a fewer number of edges points means a fewer number of slopes on the accumulator and this should make maximum search in the accumulator more accurate. We are now ready to apply the line detection process with Hough transform.

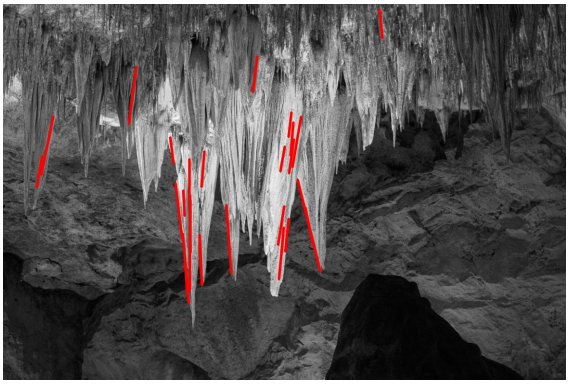


Fig. 5: Line detection after image processing.

The lines belonging to the most prominent stalactite finally appears on the image. We could actually see more lines by reducing the threshold level, but we would risk taking account of a few remaining lines created by the little noise still present on the preprocessed image, plus in our case, we don't really care about the number of stalactites but rather their presence on the camera image.

4. Visualization of the accumulator

As we pointed out earlier, it is possible to see the slopes in the parameters space : the accumulator is indeed a 2D matrix which can be read as an image with OpenCV2. Let's visualize the accumulator of the edge points with the Canny edge detector and with our own image processing.

As we were expecting, less slopes means a clearer accumulator: intersection zones are spikier and stand for the line we are actually looking for. Moreover we could wonder why do lines appear on our Canny edge detection and yet, are not detected using Hough transform. The answer is quite simple : areas of noise can form a large number of lines, the larger the noise

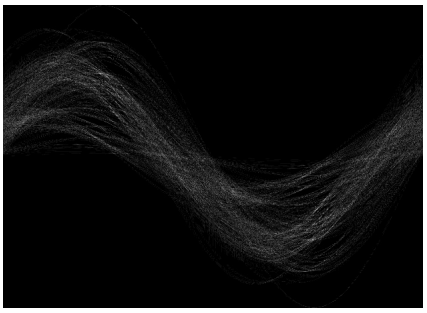


Fig. 6: Accumulator with our image preprocess.

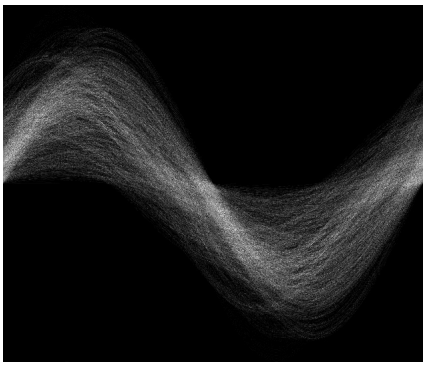


Fig. 7: Accumulator with a Canny edge detection.

area is, the longer the produced lines are. As a consequence, a line for instance 20 pixels long can have a lower score than lines produced by a 100 pixels large noise area. This is why increasing the threshold in the first place did not change the results and this is also why we observe so much horizontal lines where noise is the most present.

5. Possible improvement and use in a real time context

The hypothesis we took before about the orientation of our robot will practically never be satisfied. However, the euler angle that will modify the rho angle of lines on the camera image is the roll. As a consequence, if our robot is equiped with an IMU or any device able to measure its euler angles, we can apply the camera image a rotation equal to the opposite of our robot's roll angle. One crucial thing is that the results produced by our methods vary a lot depending on the image we look at, as the number of lines found will never be the same from on image to another. One solution would be to keep the 10% best by adapting the threshold in the accumulator. But still, it is possible for the robot to look at anything but stalactite and still find hundreds of lines.

6. Conclusion

In this paper, we suggest an image processing technique in order to produce better results of line detection on stalactites using Hough transform. The image processing produces an edge points list representing significant vertical edges. We also notice how image noise can produce edges which will impact the accumulator and produce lines out of nowhere. This subject highlights the fact that shape detection techniques are never direct and require most of the time an image processing.

6.1. Citations and references

REFERENCES

[1] Hough, P. V. C. (1959). Machine Analysis of Bubble Chamber Pictures. University of Michigan.

[2] Duda, R. O., & Hart, P. E. (1972). Use of the Hough Transformation to Detect Lines and Curves in Pictures. *Communications of the ACM*, 15(1), 11-15.

Trajectory Tracking Control in Sliding Mode and Feedback Linearization of a Mobile Manipulator

Gwendal Crequer, Student, ENSTA Bretagne

***Abstract*—This report presents a trajectory tracking control using sliding mode and feedback linearization for a redundant mobile manipulator, avoiding singularities and relying on numerical estimates. Both articulated arm and mobile robot control are considered simultaneously.**

I. Introduction

The modeling and control of polyarticulated systems, as well as that of mobile robots, are well-mastered independently today. Various methods exist, especially to avoid manipulator arms encountering singularities, whether structural or kinematic, in both control [7] and trajectory planning methods [4]. Regarding mobile robot control, it is also well-established, both in terms of control and trajectory planning. However, the formalisms and practices associated with these domains are often distinct. Current research actively explores numerous solutions for the control of mobile manipulators, typically by planning a trajectory for the mobile base and another for the robotic arm [6]. This article presents a control in sliding mode and feedback linearization of a redundant mobile manipulator (unicycle model with an *RRR* arm) for end-effector trajectory tracking. Two methods have been implemented, inspired notably by Wand et al. [3], aiming to always maximize the distance between the current configuration and singularities. The first method modifies a singular value of the Jacobian matrix when approaching a singularity, while the second applies a repulsion force based on the gradient of a criterion with respect to the controls.

II. Modeling of Polyarticulated Mobile Robots

The modeled robot is an *RRR* arm mounted on a unicycle-type mobile base. The unicycle robot is modeled by a state vector $\mathbf{x} = [x, y, \theta]^T$, where x and y are the Cartesian coordinates of the robot, and θ is the angle of the mobile base with respect to the z axis, perpendicular to the horizontal plane. The *RRR* arm is modeled by a state vector $\mathbf{q} = [q_1, q_2, q_3]^T$, where q_1, q_2, and q_3 are the angles of the arm joints (Khalil-Kleinfinger representation in Table I).

TABLE I
Khalil-Kleinfinger Parameterization for *RRR* Arm

	α	**d**	**r**	θ
${}^0\mathcal{T}_1$				q_1
${}^1\mathcal{T}_2$	$\frac{\pi}{2}$		l_1	q_2
${}^2\mathcal{T}_3$		l_2		q_3

The pose of the mobile platform is represented by the homogeneous transformation matrix ${}^b\mathcal{T}_0$ as follows:

$$ {}^b\mathcal{T}_0(\mathbf{x}) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 & x \\ \sin(\theta) & \cos(\theta) & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1} $$

The direct geometric model of the *RRR* arm, providing the position of the end effector, is given by the following equations:

$$ {}^0\mathbf{P} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \overbrace{\underbrace{\begin{pmatrix}\mathbf{I}_{3,3} & \mathbf{0}_{3,1}\end{pmatrix}}_{\text{Projection}} \underbrace{{}^0\mathcal{T}_1(q_1)\,{}^1\mathcal{T}_2(q_2)\,{}^2\mathcal{T}_3(q_3)}_{\text{Homogeneous Transformations}}}^{\mathbf{g}(\mathbf{q})} {}^3\mathbf{P} \tag{2} $$

in the base mobile coordinate system. Thus, the position of the end effector in the world frame can be obtained by the relation:

$$ {}^b\mathbf{P} = \underbrace{\begin{pmatrix}\mathbf{I}_{3,3} & \mathbf{0}_{3,1}\end{pmatrix}\ {}^b\mathcal{T}_0(\mathbf{x})\ {}^0\mathcal{T}_1(q_1)\ {}^1\mathcal{T}_2(q_2)\ {}^2\mathcal{T}_3(q_3)\ {}^3\mathbf{P}}_{\mathbf{h}(\mathbf{x},\mathbf{q})} \tag{3} $$

A. Kinematic Model

The kinematic model of the mobile base is given by the following equations:

$$ \dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{pmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{pmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{4} $$

with v and ω as the linear and angular velocities of the mobile base, which are the robot controls. Here, the main reason for choosing the unicycle model is its linearity in input dependencies on the observed output, similar to articulated robots.

The kinematic model of the *RRR* arm is given by the following equations:

$$ {}^0\dot{\mathbf{P}} = \mathbf{J}_{arm}(\mathbf{q})\ \dot{\mathbf{q}} \tag{5} $$

With $\mathbf{J_{arm}}(\mathbf{q})$ as the Jacobian matrix of the function $\mathbf{g}$ and $\dot{\mathbf{q}}$ as the angular velocities of the joints, which are the arm controls. As both models have input linear dependencies on the first derivative of the desired output, it is possible to combine them to obtain a complete kinematic model of the robot. By letting $\mathbf{U} = \left[[v\ \omega]^T\ \ \dot{\mathbf{q}}^T\right]^T$, and $\mathbf{X} = \left[\mathbf{x}^T\ \mathbf{q}^T\right]^T$, the following model is obtained:

$$ {}^b\dot{\mathbf{P}} = \mathbf{J}_{robot}(\mathbf{X})\ \mathbf{B}(\mathbf{X})\mathbf{U} \tag{6} $$

With $\mathbf{J}_{robot}(\mathbf{X})$ as the Jacobian matrix of the function $\mathbf{h}$ and $\mathbf{B}(\mathbf{X})$ as the matrix explicitly defining $\dot{x}$ and $\dot{y}$ from v. This method is adapted from that proposed in [2].

III. Robot Control

Now that we have a system of equations of the form $\dot{\mathbf{y}} = \mathbf{AU}$, we can define the control $\mathbf{U}$ to achieve the desired trajectory tracking [1]. Using feedback linearization control, the control $\mathbf{U}$ can be defined as follows:

$$\mathbf{U} = \mathbf{A}^{-1}\mathbf{v} \tag{7}$$

where $\mathbf{v}$ is the first derivative of the trajectory to be adopted to perform the task. Since the matrix $\mathbf{A}$ is not square (redundancy), its pseudo-inverse must be used.

A. Feedback Linearization Control

The differential equation to pose to converge the trajectory error for a system with a relative degree of 1 is given by:

$$\underbrace{(\dot{\mathbf{y}}_d - \dot{\mathbf{y}})}_{\dot{\mathbf{e}}} + \frac{1}{\tau}\underbrace{(\mathbf{y}_d - \mathbf{y})}_{\mathbf{e}} = 0 \tag{8}$$

One can then choose $\mathbf{v} = \dot{\mathbf{y}} = \dot{\mathbf{y}}_d + \frac{1}{\tau}\mathbf{e}$ to achieve convergence of the trajectory error in a characteristic time τ.

B. Sliding Mode Control

The principle of sliding mode control is not to use a *PD*-type correction to achieve convergence of the trajectory error but to force the system to stay on a sliding surface that is no longer linearly dependent on $\mathbf{U}$ (surface of $\left\{e^{(k)}\right\}$ where the differential equation converges to 0). For this, the differential equation to be respected on the error is defined as:

$$\mathbf{e} = 0 \tag{9}$$

Because our Jacobian-based linearization led to a first-order system. The control can then be chosen as follows to quickly converge to a sliding surface:

$$\mathbf{U} = \mathbf{A}^{-1}\left(K \times \text{sign}(\mathbf{e})\right) \tag{10}$$

With K large to converge quickly. Then, there are some tricks to decrease the value of K and avoid *chattering* effects.

IV. Results with Classical Commands

Figure 1 depicts a unicycle with an articulated arm designed to follow a given trajectory. Implementation of commands using *feedback linearization* and *sliding mode* allows for the evaluation of the distance between the end effector and the target. Convergence towards the desired position is indeed confirmed, but certain situations lead to a sudden increase in error and state parameter lock-ups. This occurs during the traversal of a singularity, either structural or kinematic, in the robot.

The traversal of these singularities results in the non-invertibility of matrix $\mathbf{A}$, which can be characterized by the determinant of $\mathbf{A}^T\mathbf{A}$ or its singular values. All these metrics are presented in Figure 2.

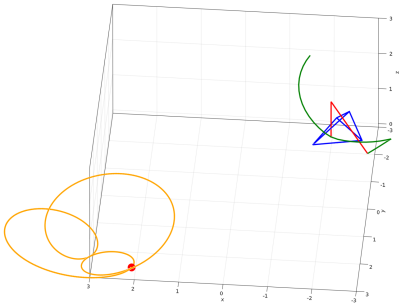


Fig. 1. Mobile manipulator and trajectory to follow

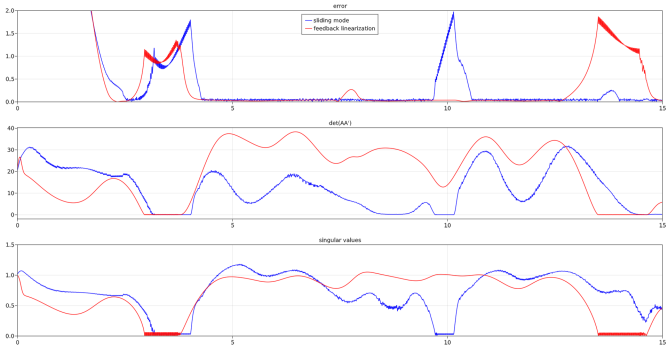


Fig. 2. : Distance to the target, determinant, and smallest singular value. In blue: Sliding mode, in red: Feedback Linearization

V. Singularity Avoidance

To avoid singularities, several solutions exist, notably explained in the case of articulated arms by Wand et al. [3]. They propose 2 methods:

- Decompose the matrix $\mathbf{A}$ into singular values and artificially increase the value of the smallest one, even if it means exiting the sliding surface.
- Use repulsion based on the gradient of a criterion (determinant or singular value) to prevent the system from approaching the singularity.

It is worth noting that, at least for now, these methods are applied after the definition of the command. A discussion on this will be held in the conclusion.

A. Artificial Increase of Singular Value

The first method involves inverting matrix $\mathbf{A}$ using its Singular Value Decomposition (SVD) and modifying the value of the singular values at the time of inversion.

$$\begin{cases} \mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \\ \mathbf{A}^{-1} = \mathbf{V}\mathbf{\Sigma}^{-1}_{\text{modified}}\mathbf{U}^T \end{cases} \tag{11}$$

With:

$$\mathbf{\Sigma}^{-1}_{\text{modified}} = \text{Diag}\left(\frac{\sigma_i}{\sigma_i^2 + \lambda^2}\ldots\right) \tag{12}$$

such that when σ_i is large, the inverse is close to $\frac{1}{\sigma_i}$, and when σ_i is small, the inverse is close to $\frac{1}{\lambda}$, with λ being a regularization parameter.

B. Repulsion

The second method involves estimating the gradient of the determinant of $\mathbf{A}^T\mathbf{A}$ or the smallest singular value for a small variation in commands, and applying a repulsive force proportional to this gradient to choose a command that moves away from singularity.

$$\mathbf{U} = \mathbf{A}^{-1}\mathbf{v} + \alpha \times \nabla_{\mathbf{U}}\left(\varphi(\mathbf{X})\right) \tag{13}$$

With $\varphi(\mathbf{X})$ being the criterion to minimize, $\varphi(\mathbf{X}) = \det\left(\mathbf{A}^T\mathbf{A}\right)$ or $\min\left(\mathbf{\Sigma}\right)$, and α being a regularization parameter.

VI. Simulation and Results

The Jacobian and gradient calculations rely on numerical estimates based on a second-order centered finite difference calculation:

$$\begin{cases} \nabla_{\mathbf{U}}\left(\varphi(\mathbf{X})\right) = \frac{\varphi(\mathbf{X}+\delta\mathbf{U})-\varphi(\mathbf{X}-\delta\mathbf{U})}{2\delta} \\ \mathbf{J}_{\text{robot}}(\mathbf{X}) = \frac{\mathbf{h}(\mathbf{X}+\delta\mathbf{X})-\mathbf{h}(\mathbf{X}-\delta\mathbf{X})}{2\delta} \end{cases} \tag{14}$$

Using the same pattern as Figure 1, the results obtained with singularity avoidance methods are observed in Figures 3 and 4. Kinematic saturations have been added to involve kinematic singularities in the robot's behavior.

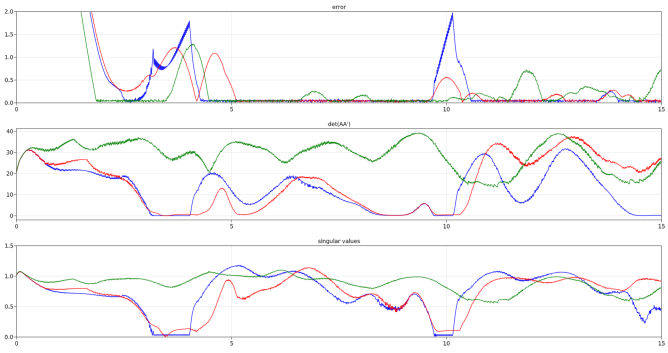


Fig. 3. Sliding mode: Distance to the target, determinant, and smallest singular value. In blue: without avoidance method, in red: with modified singular value, in green: with repulsion method

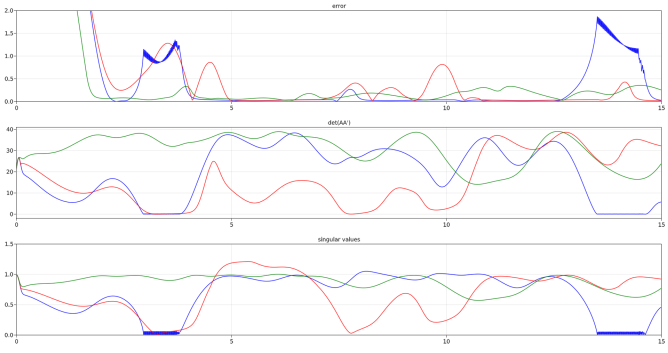


Fig. 4. Feedback Linearization: Distance to the target, determinant, and smallest singular value. In blue: without avoidance method, in red: with modified singular value, in green: with repulsion method

The most notable result is the use of repulsive potential coupled with feedback linearization, where the overshoot due to trajectory error is 4 times smaller than without avoidance method (from $2ua$ to $0.5ua$ in singular situations), and 3 times smaller than the artificial increase in singular value method. Error increases without determinant cancellation can also be explained by three reasons:

- The command overshoot (nonexistent here), due to the system order, is indicative of feedback linearization control.
- The two presented methods modify the command, making it no longer a solution to the control problem.
- Some singularities are not avoided by these methods, especially kinematic singularities. They must be considered in planning as they are not locally detectable (a local minimum can be too far from the upcoming solution).

It is also necessary to consider that the avoidance method parameters are the same for both commands, and the parameters of these commands have not been modified either. This choice was made considering that the corrections came in addition to the command, but it could also be entirely possible to compare the results by always seeking the best control and avoidance parameters.

VII. Conclusion

The control and formalism of mobile robots can be applied to polyarticulated robots with a conversion of the formalism of robotic arms. However, this fusion implies considering the kinematic and structural singularities of the arm, workspace limitations, and other constraints that do not exist for mobile robots. Moreover, in the present case, the use of a unicycle model was very simplifying, as it is a model whose inputs are linearly dependent on the output, similar to robotic arms. Thus, the use of the Jacobian was well-suited. For other more complex robots, it is advisable to reconsider this simplification and work analytically on the state equations to find the relative degrees of each input and add differential delays if necessary on the inputs [1].

Finally, regarding the implementation of commands, they were added roughly as corrective terms. It could be possible, especially for sliding mode control where the goal is to reach a sliding surface, to calculate and transform the corrections to keep them on the surface, thus making them a solution to the control problem. This would mean, for example, in the case of adding repulsion, not adding it to the command but calculating the gradient in the kernel of the application to find a command that stays in the kernel, and thus a solution.

References

[1] S. Buss, "Introduction to inverse kinematics with Jacobian transpose, pseudoinverse and damped least squares methods," IEEE Transactions in Robotics and Automation, 2004.

[2] M. Boubezoula, "Modélisation et commande d'un manipulateur mobile à roues," Ferhat Abbas University, 2019.

[3] B. Bayle, "Mod elisation des manipulateurs mobiles a roues, en vue de leur commande cinematique," 2001.

Rapidly-Exploring Random Trees

Apolline de Vaulchier du Deschaux

In this article, the concept of the Rapidly-Exploring Random Tree (RRT) is explained. The description of this path planning algorithm (based on tree incrementation) and its associated parameters is introduced, and their influence is experimentally examined to facilitate a discussion of the obtained results. The algorithm is applied to the scenario of two arms holding an object (2D simulation). The objective is to move the object from point A to point B while avoiding an obstacle. The RRTs aspect is also mentioned, with path optimization by calculating a cost over the distance between nodes.

RRT | RRTs | Dual-arm | Path Planning
Correspondence: ***apolline.de_vaulchier@enstabretagne.org***
Git : ***https://gitlab.ensta-bretagne.fr/devaulap/myrtt.git***

Introduction

RRT were introduced by Steven M. LaValle in 1998 in his article named "Rapidly-Exploring Random Trees: A New Tool for Path Planning" (1). The main idea behind RRT was to create a path-finding algorithm capable of efficiently handling high-dimensional configuration spaces, often encountered in the context of trajectory planning for mobile robots and autonomous systems. The tree is built incrementally in order to quickly reduce the expected distance between a randomly selected point and the tree. The tree is built randomly and biased to reach all the unexplored areas of the problem, it can be associated with a Voronoi diagram.

The advantages of RRT are its adaptability to complex environments, rapid exploration of a space, robustness in the face of dynamic changes, and ease of implementation. On the other hand, we need to know the environment, and it doesn't guarantee that you'll find an optimal path. For that, you need to add a best-path search function. It is also sensitive to its initial parameters, such as branch size or number of nodes. Mr. LaValle also introduces the concept of RRTs, which corresponds to the RRT algorithm with path optimization. The cost principle for each path is computed per branch. This cost can be calculated in various ways, such as the distance between each point, the angle between each position, or the energy expended for each configuration. B. An and al., in their article (2), mention the calculation of a maximum position deviation of an open chain arm, thus limiting the gap between each tree node.

The goal of this article is to apply the RRT to two 2-degree-of-freedom arms holding the same object. The object needs to move from point A to point B in a 2D space. In this known space, an obstacle is positioned. Optimization of the found paths is then carried out, forming an RRTs (s for star). Indeed, RRT's algorithm was first used in 1998. Since then, many improvements and versions of this algorithm have been made. The aim of this article is to introduce the basics of RRT, not to develop or explain its optimization.

Thus, Section 1 discusses the operation of the RRT algorithm and its associated parameters. Section 2 describes the Dual-arm project and the implemented algorithm. Section 3 details the obtained results, and Section 4 discusses the various outcomes.

1 RRTs algorithm

A. Notation.

The following notations will be used in the next section to explain the RRT algorithm.

X: Space

$C(X)$: Space and robot constraints

x_{init}: First node of the tree

x_{rand}: Random node that doesn't belong to the tree

x_{near}: Node closest to x_{rand}

x_{new}: Node created with x_{rand} and x_{near}

ϵ: Maximum distance parameter associated with RRT

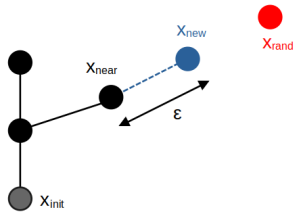


Fig. 1. How to add a node to a tree and form a new branch.

B. Algorithm.

The first node in the tree corresponds to x_{init}, which is the robot's initial position. Next, a node x_{rand} in space X is randomly selected and the node x_{near} in the tree closest to this node is found. A new node x_{new} is calculated on the right-hand side between x_{rand} and x_{near}, with distance ϵ like in the figure 1. If the distance between the two nodes is less than ϵ, then x_{new} corresponds to x_{rand}. The node is added to the tree only if it does not belong to the $C(X)$ constraint of the environment. This gives control over the size of the incremental growth. You can also bias the growth of the tree to guide the search to specific areas of the space. By putting a low probability on certain cells and higher probabilities on the cells of interest then the tree will grow much more in the

cells with higher probabilities. Then we repeat the process as many times as we wish to have vertices K. Once you have obtained this roadmap via the RRT, you can proceed to search for the most optimal path and form the RRTs.

The idea behind RRTs optimization is to take into account a parameter that enables nodes from other branches to be linked. This cost parameter differs according to the methods chosen and the constraints imposed by the robot. The idea is to form a circle of radius r around the newly created node and look at the nodes present in this circle. If the calculated cost is less than the distance between the new node and the previous node on its branch, then a new link is created between them. This principle is illustrated in figure 2.

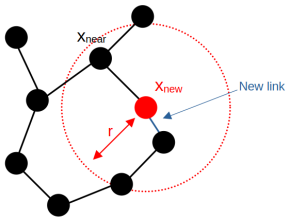


Fig. 2. Principe of RRTs.

2 Dual-Arm's project

A. Objective.

For this project, the motion planning of a dual-arm robot holding an object in its hands in 2D is studied. Each arm of the robot is composed of 3 links linked by Rpivot. The goal is therefore to study the closed chain formed by the two arms holding this object to move it from a point A to a point B. Python software was used to carry out the obtained algorithm of RRTs and results.

B. Problem definition and notations.

My space X is defined by the x axis and the y axis. In this space there is a rectangular obstacle defined by $O_b = (x_{ob}, y_{ob}, w, h)$ where x_{ob} is its x axis coordinate center, y_{ob} is its y axis coordinate center, w its width and h its height.

There are two Rpivot arms with 2 links each over. The hands of each arm are linked to the object. Together they form a fifth link. Each link is identified by L_i for $i = 1,2,3,4,5$. The position of each pivot is named Pos_{pi} for $i = 1,2,3,4,5,6$. The representation is on figure 3.

There are two frames: the fixed frame $\{0\}$ and the object frame $\{B\}$ in the middle of the object held by the robot. This one is defined by (P_x, P_y, ϕ) which represent the distance between the two frames and the angle between the object frame and the fixed frame. The object is held in a completely fixed way and the two hands are aligned so the size of the link between the pivot Pos_{p3} and Pos_{p4} depends on the width of the object between the two hands.

For this problem, we suppose that $L = L_1 = L_2 = L_4 = L_5$ and $L_3 = width_{object} + L$. Moreover, the position of the Rpivots at the beginning of each arm is fixed. This Dual-arm configuration leads to the retention of a closed chain. This closed chain has four possible configurations for an object position. So, for each node of the tree, all 4 configurations will be taken into account.

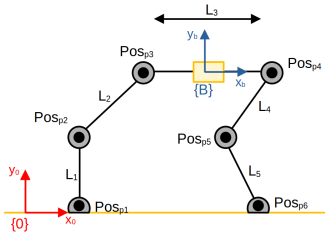


Fig. 3. Definition of the Dual-arm's problem.

The following notations will be used in the next section to explain the algorithm:

q: representation of a node in the algorithm

- $q.coor$: position and orientation of the node (q_x, q_y)
- $q.cost$: cost of the node
- $q.parent$: id of its parent's node in the tree
- $q.pos$: list of available configuration of the node

For example, q_{strat} represent the initial node of the tree with $q_{start}.coor = x_{init}$, $q_{start}.cost = 0$, $q_{start}.parent = 0$, $q_{start}.pos = (1,2,3,4)$.

C. Algorithm.

There are two parts in this algorithm. The first is the one to get the roadmap via the RRT. The second uses this roadmap to find the best path among all those obtained (the RRTs part). In order to improve the program, these two parts are incorporated together in my algorithm. The first step is to choose a random point in space. Then, it is to check that this point is in the workspace of the robot using *InWorkSpace* function. This one takes as argument $q.coor$, the position of the two fixed pivot and the length L and L_3 . In order to check if the point is in the workspace the intersection between the two circles formed by Pos_{p3} and Pos_{p1} of radius L each and the two circles formed by Pos_{p4} and Pos_{p6} of radius L each is check. If for both sides there is an intersection then the point is in workspace as figure 4 shows.

Once q_{rand} has been obtained, we must find the point in the tree closest to q_{rand} , q_{near} . To do this, the distance between all the points in the tree and $q_{rand}.coor$ is evaluated and take the minimum obtained is taken. Then a new point is created from q_{rand} with the *NewPoint* function which takes as argument the distance between q_{rand} and q_{near} , these two points and ϵ which corresponds to the maximum distance. If the distance between the two points is less than ϵ then $q_{rand}.coor = q_{new}.coor$. Otherwise $q_{new}.coor$ is the

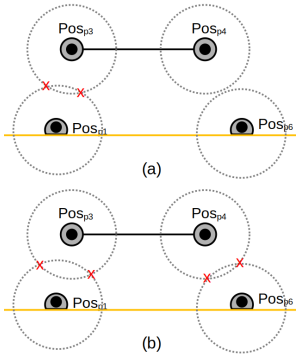


Fig. 4. (a): Configuration which is not in robot workspace. (b): Configuration which is in robot workspace.

maximum distance ϵ in the direction of q_{rand} . Then we check that q_{new} is still in the workspace with *InWorkSpace* function. Before adding this point to the tree, this point must not represent a configuration of the robot collision with the obstacle. For this, two functions is used: *FindPoint* and *Collision*. *FindPoint* allows you to find all the configurations of the robot for a position (P_x, P_y, ϕ). Indeed, there are 4 of them. Again, the intersection of the circles to get all the solutions is employed. Once all the positions of the pivots are obtained for the position of the object, each solution is tested to check that each of them does not collision with the obstacle. The *Collision* function is used, it takes as argument the list of solutions, the robot parameters and the position of the obstacle. The goal is to create two polygons and check that they do not intersect. The first polygon is made from the data on the obstacle and the second is made from the Pos_{pi} and the shape of the object. If there is no collision, add the point to the tree and repeat the operation as many times as you wish to have points in the tree. The outputs of function *Collision* are a Boolean to indicate collision and a list of possible configurations.

Now the tree is built, let's find the best way. This part occurs between getting q_{new} and adding it to the tree. It allows to associate to each position q its cost and its least expensive parent. The cost of a point corresponds to the Euclidean distance between q and the nearest point q_{near}. Once this cost has been assigned, all the points in the tree that are within a radius r of this point q_{new} are selected. Then, a comparisons between all these points obtained is done to find out if the cost of one of these points plus the distance between it and q_{new} is less than the current cost of $q_{near}.cost$ plus the distance between them. The point which costs the least is then associated as a parent of q_{new}. Once all the points of the tree are obtained, we just have to go back up the list of parents between q_{goal} and q_{start} to obtain the optimal path.

3 Results

Here 5 are the results obtained with this algorithm for a starting point $A = (25, 30, 0)$ and an end point $B = (50, 20, 0)$. The max number of node to choose is 9500, $\epsilon = 1$ and $r = 3$. The object is a pencil (less width than arms). The black lines represent the tree and the blue lines are all the paths that can be made in this tree. The red line represents the optimal path found.

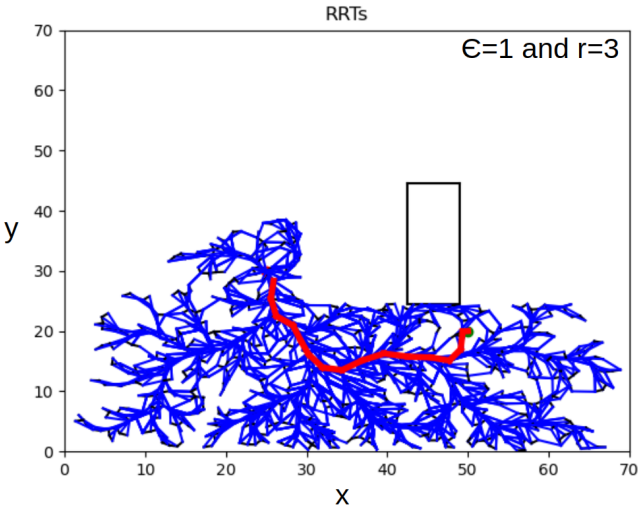


Fig. 5. RRTS for $\epsilon = 1$ and $r = 3$.

The algorithm indicates, for this problem, that two configurations are possible for the optimal path. This is confirmed with the plot of the different configurations for each node of the optimal path cf. figure 6. Configurations 2 and 4 are those which do not collide with the obstacle.

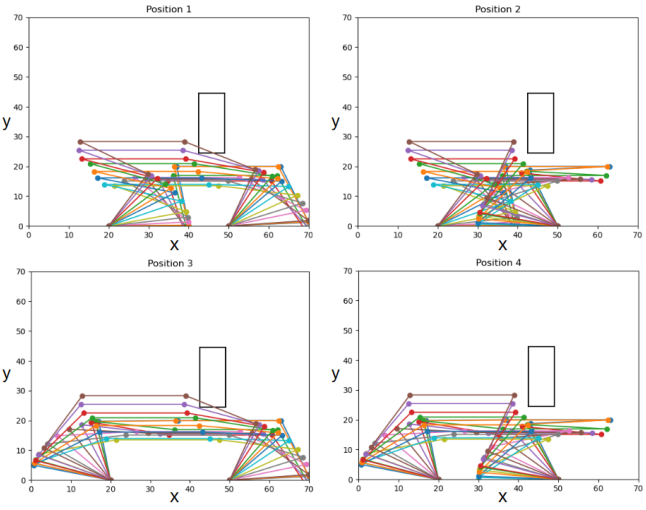


Fig. 6. The four configurations for the optimal path with $\epsilon = 1$ and $r = 3$.

The tree forms well around the obstacle. There are empty areas at the top of the environment. This corresponds to the first of the reachable zones due to the shape of the robot and the second of the unreachable zones due to the obstacle. Taking into account the different configurations seems coherent. Path optimization seems to be effective with regard to the choice of start and end points, with both arms taking the shortest route. On the other hand, they seem to go too far down the obstacle in relation to the arrival point. This is probably due to the arbitrarily chosen parameters ϵ and r.

Once the model has been validated, the influence of parameters ϵ and r is studied. Figure 7 shows the RRTs for

$\epsilon = 0.01$	$\epsilon = 0.1$	$\epsilon = 1$	$\epsilon = 10$
18.0	11.2	11.4	12.4
$r = 0.05$	$r = 1.5$	$r = 3$	$r = 10$
1.0	1.5	11.4	13.6

Table 1. Process time in second

a fixed $r = 3$ and $\epsilon = 0.01, 0.1, 1, 10$. Conversely, figure 8 shows RRTs for fixed $\epsilon = 1$ and $r = 0.05, 1.5, 3, 10$.

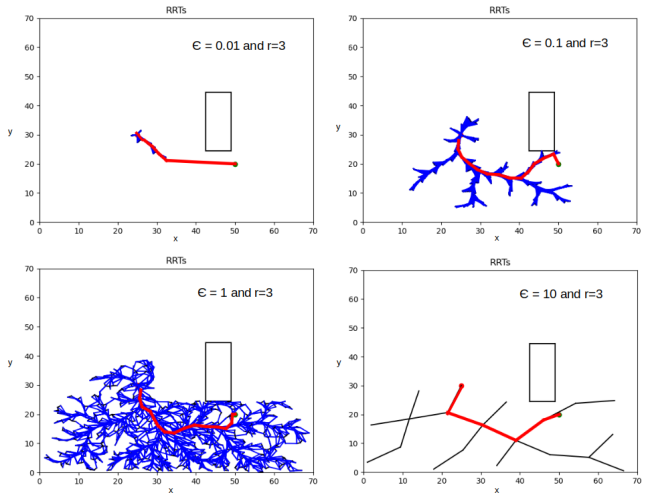


Fig. 7. The four results for $\epsilon = 0.01, 0.1, 1, 10$ and $r = 3$.

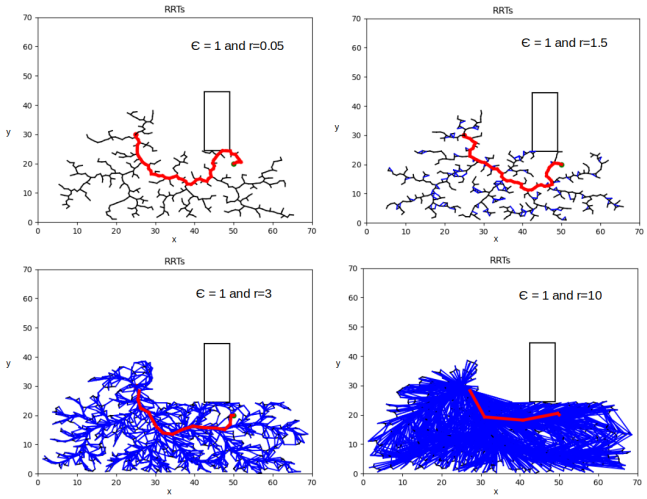


Fig. 8. The four results for $r = 0.05, 1.5, 3, 10$ and $\epsilon = 1$.

For each simulation, the process time is computed. Strictly speaking, this does not necessarily have any significance. Indeed, the algorithm used has not been optimized, and the process time of an RRT depends on the complexity of the environment and the number of constraints on the robot. On the other hand, it may be interesting to know whether modifying the parameters e and r has an influence on the process time. These times should be taken with precaution, as they depend on the choice of random point (the solution is reached more or less quickly depending on the random part). The results obtained are on table 1.

4 Discussion

In this algorithm there are two factors that are randomly chosen: the maximum distance ϵ and the radius r. By increasing the maximum distance ϵ the tree is less and less developed as figure 7 shows. On the contrary, if it decreases too much ϵ then the branches of the tree become too small and it is not possible to form different paths between the points. A study was carried out by B. An and al (2). to determine the optimal distance between each point . By exploiting the in joint configuration space standards and the forward kinematics of the robot, they obtain an appropriate angle delta to move each joint and thus an appropriate delta to move the end-effector while respecting the robot constraints. The algorithm has been made for an open chain but can be modified to apply it to a closed chain. Indeed, A possible improvement of the algorithm is to compute for each arm the parameter of B. An and al. and then averaged of them will give the final maximum distance.

If this time it is r which vary then it increase or decrease the number of paths between the points of the tree. If r is very large, then you can move from one position to another very distant position. This is no longer appropriate if the dynamic aspect is add to this study. Otherwise, you cannot go through any point except the one created by the tree. This restricts a lot the possibilities of path and we do not obtain a very optimal path.

Finally, process times are consistent with the influence of parameters ϵ and r on RRTs. For a fixed r, time forms a kind of parabola as a function of ϵ. The choice of ϵ could then be based on the minimum of this parabola. Then, for a fixed ϵ and varying r, the time function appears to be increasing. On the other hand, if r decreases too much, then the choice of the optimal path is no longer very developed. We therefore need to choose a middle ground between the two, or change the algorithm to find the optimal path as in the article of I. Noreen, A. Khan and Z. Habib (3) with their RRTs-Smart. It runs similarly to RRTs, but performs a path optimization process when an initial path is found. This optimization process eliminates redundant nodes from the initial path found. In addition, it also identifies beacon nodes for path improvement. The second major feature introduced by RRTs-Smart is intelligent sampling. This differs from random sampling in that it is directed towards the tag nodes of the optimized path.

In this study only the aspect of the position of the object is taking into account. The cost attributed to the travels corresponds to the distance between each point. It do not take for example the difference between the two angles phi of these points. If we added the dynamic aspect of the problem, then it would be interesting to calculate the cost of a movement by compute the energy needed to move from one position to another one, without having to compute the Euclid distance. And thus, it will take into account the inclination between these two points.

Bibliography

1. Steven M. LaValle. Rapidly-Exploring Random Trees: A New Tool for Path Planning. *Iowa State University*, October, 1998.
2. B. An and al. An Adaptive Stepsize RRT Planning Algorithm for Open Chain Robots. *IEE Robotics and Automation Letters*, 2017.
3. Amna Khan Iram Noreen and Zulfiqar Habib. A Comparison of RRT, RRT* and RRT*-Smart Path Planning Algorithms. *International Journal of Computer Science and Network Securit*, October, 2016.

Drone recognition by Micro-Doppler signatures

Martin Galliot

In this article, the application of Micro-Doppler signatures for drone detection will be simulated. It consists in a frequency modulation created by the intern movements of the observed target, in this case, the spinning propellers. The project focuses on generating simulated datasets of drone's Micro-Doppler signatures for further uses like deep learning algorithms.

Unmanned aerial vehicles | Drone | Micro-Doppler | Radar detection | Recognition

Correspondence: ***martin.galliot@enstabretagne.org***

Introduction

Over the years, drones, also referred to as unmanned aerial vehicles (UAVs), have witnessed a surge in popularity owing to technological advancements and their diverse applications across various industries. One of the key drivers behind the increased utilization of drones is their capability to undertake tasks that are challenging, hazardous, or unfeasible for humans. As technology progresses and production costs decrease, it is foreseeable that drones will emerge as an increasingly indispensable tool across a multitude of industries and applications.

Nevertheless, as well as polluting air traffic, their widespread adoption has also sparked apprehension regarding their potential for weaponization or malicious intent. A primary concern surrounding drones is their susceptibility to exploitation in terrorist activities. Equipped with cameras, explosives, or other armaments, drones possess the capacity to surveil, infiltrate secure zones or execute assaults. This renders them an enticing prospect for terrorist entities due to their affordability, accessibility, and remote operability. Furthermore, their compact size and ability to fly at low altitudes render them elusive and challenging to intercept, amplifying their threat to security.

Given these concerns, governments and various entities are initiating measures to oversee and control drone usage, such as maximum authorized altitudes, mandatary licenses and registered drones. However, despite these endeavors, drones persist as a significant threat. It is therefore essential to detect and classify unidentified drones in the context of terrorism and conventional war. The goal of this article is to see if the Micro-Doppler effect is relevant to do that and to generate datasets of these signatures.

Thus, Section 1 describes Doppler and Micro-Doppler effect. Section 2 uses a MATLAB simulation to generate datasets of drones's Micro-Doppler signatures. Section 3 discusses the various outcomes and further uses of these datasets.

1 Doppler and Micro-Doppler Effect

A. Phenomenon.

The Doppler effect (1) is a phenomenon observed when a wave source, such as sound or light, moves in relation to an observer. This results in a shift in the frequency of the wave as detected by the observer. The equation for the Doppler effect is given by :

$$f_d = \frac{v_s \pm v_o}{v_s} \cdot f_s$$

where f_d is the observed frequency, f_s is the frequency of the source, v_s is the velocity of the source, and v_o is the velocity of the observer. The sign of v_o depends on whether the observer is moving towards (-) or away from (+) the source of the wave.

The equation comes from the relationship between wave frequency, velocity, and wavelength. When source and observer move relative to each other, it alters the observed frequency. For instance, as a car approaches an observer while honking its horn, the perceived frequency increases due to sound wave compression. This Doppler effect finds various applications, including radar, medical imaging, and astronomical velocity measurements.

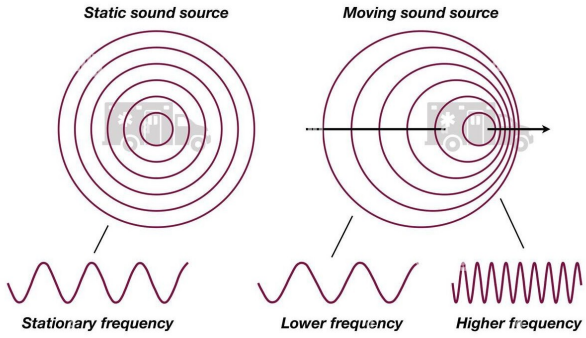


Fig. 1. Doppler effect (2)

The Doppler effect is frequently employed for detection, with one of its primary benefits being the extensive range achievable by Doppler Radar.

B. Limitation of Doppler effect.

While Doppler pulse radar can detect objects at very high ranges, drones typically have smaller Radar Cross-Sections (RCS) compared to helicopters or fighter jets, making them more challenging to identify. Existing airspace monitoring methods are primarily tailored to detect faster and higher-altitude objects than mini-UAVs. For example, AWACS radars intentionally filter out slow-moving targets like birds

to reduce false alarms. Domestic military radars likely employ similar algorithms. Consequently, a mini-UAV flying at low altitudes and speeds might register on radar as a bird rather than a potential threat like a cruise missile or fighter jet. Moreover, air defense radar may struggle to detect mini-UAVs flying low over urban areas due to signal scattering from larger background objects. Hence, further research is necessary to discern the specific factors affecting radar detection of mini-UAVs.

One solution to these limitations could be to use micro-Doppler effect which allows a more detailed analysis of the Doppler shift.

C. Micro-Doppler Effect.

The micro-Doppler effect (3) describes the frequency modulation of a radar signal induced by the oscillatory movement of an object or its structural components. This motion can arise from various factors, such as propeller or rotor blade rotation, bird wing flapping, or human limb swinging. When a radar beam interacts with an object exhibiting micro motion, the frequency of the reflected signal undergoes a Doppler-induced shift, generating frequency modulation containing harmonic frequencies. This modulation can be leveraged to ascertain the object's kinematic characteristics and facilitate target identification. Despite the long-standing proposal of using micro motion for target identification, techniques for representing the time-varying frequency modulation as a signature, extracting kinematic information, and employing the signature for target identification remain active research areas. The micro motions of a drone's body and propeller blades yield a distinct micro-Doppler signature, which can be analyzed for detection and classification purposes. This signature enables differentiation of drones from other objects, such as birds, and identification of specific drone types.

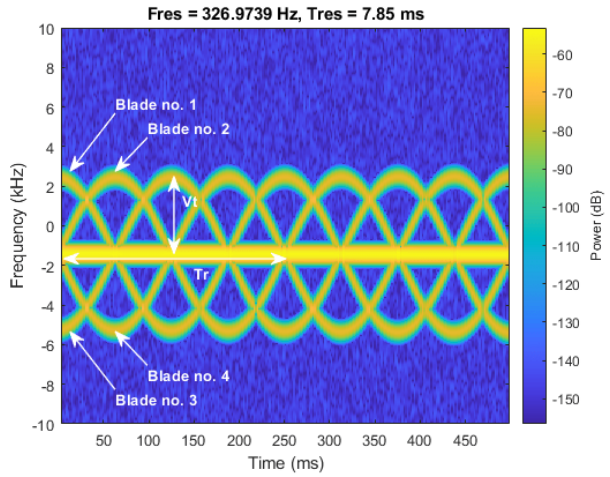


Fig. 2. Simulated Micro-Doppler modulation for a helicopter (4 blades) (4)

The figure 2 shows the micro-Doppler modulation caused by blade tips around a constant Doppler shift. The image suggests that each blade tip introduces a sinusoid-like Doppler modulation. Within each period of the sinusoid, there are three extra sinusoids appearing at equal distance. This appearance shows that the helicopter is equipped with four equally spaced blades.

2 Generation of dataset

A. Objective.

A Pulse-Doppler Radar (PDR) is a radar system designed to detect and track moving targets such as helicopters or drones. It operates by transmitting bursts of pulses, analyzing the returned pulses to extract Doppler information. The resolution of the Doppler information depends on factors such as the number of pulses in each burst and the Pulse Repetition Frequency. The objective here is to generate PDR signatures through MATLAB (4) simulation.

B. Parameters.

We chose parameters so that the generation closely mimics realistic radar and target properties (5) (6):

- A carrier frequency of 5 GHz and a Pulse Repetition Frequency (PRF) of 50 KHz.
- Each input sample in the simulation represents a Doppler signature extracted from 64 bursts of 64 pulses. This configuration ensures that there are 64 spectrums of 64 points each, allowing for the full rank of the covariance matrix to be computed over non-normalized Doppler bins.
- Target variations are accounted for by altering the number of rotating blades, with either one, two, four or six blades, as can be found on drones and radio-controlled helicopters.
- The diversity among target classes is ensured by varying factors such as receiver noise, blade size, revolutions-per-minute (RPM), and bulk speed.
- For each target class, 3000 samples are simulated, resulting in a total dataset size of 12,000 samples. While this dataset size might be considered small for deep learning tasks, obtaining thousands of relevant and labeled real radar detections would be challenging in the radar industry, making larger simulated datasets less realistic for this specific use case.

C. Results.

The dataset's quality is verified visually, ensuring easy discrimination between target classes. Here is a sample of each target class :

The classes are distinguished by the varying number of rotating blades, making the modulation pattern easily discernible. The initial set of images showcases Doppler signatures, which represent the time-varying periodogram of targets across 64 bursts of 64 pulses. In these images, each

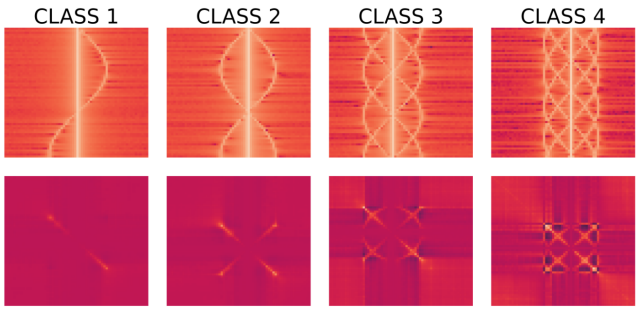


Fig. 3. Sample of each target class : one, two, four and six rotating blades

row corresponds to the periodogram computed for a single burst, while each column represents a Doppler bin. The subsequent set of images displays the covariance Spectral Power Density (SPD) representation of the samples from the initial set. Within each class, the width of the Doppler modulations around the bulk speed on the periodograms varies, as does the bulk speed itself, which is indicated by the central vertical illumination of the signature.

The bladerate can be determined by measuring the period of the corresponding sinusoid. For example, by looking at the Figure 2, $T_r = 250ms$, which implies a bladerate set at 4 revolutions per second.

The image also illustrates the tip velocity Vt, which can be deduced from the maximum Doppler shift. This maximum Doppler shift is approximately 4 kHz removed from the constant Doppler caused by the bulk movement. Thus :

$$Vt_{detect} = dop2speed(4e3, \frac{c}{f_c})/2$$

Where c is the speed of light, f_c the carrier frequency and $dop2speed$ a MATLAB fucntion that converts Doppler shift to speed.

Vt_{detect} value represents the maximum tip velocity along the radial direction. To accurately determine the maximum tip velocity, it's essential to consider the relative orientation. Since the blades spin in a circular motion, the detection remains unaffected by the azimuth angle.

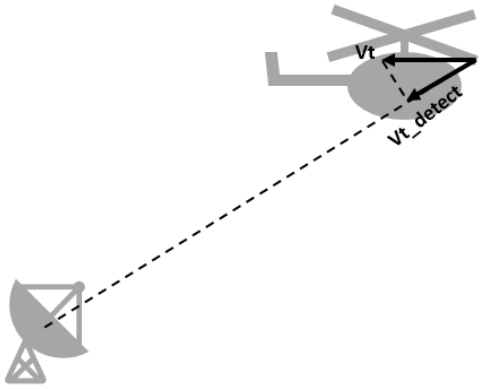


Fig. 4. Difference between Vt and Vt_{detect} (4)

3 Discussion

As mentioned earlier, we have a total dataset of 12.000 samples, which could be considered as small for deep learning tasks. However, larger simulated datasets would be less realistic because having thousands of relevant and labeled real radar detections would not be trivial in the radar industry. Firstly, most radars are not tailored for detecting small targets like mini-drones. Secondly, even those calibrated for such targets often do not capture the micro-Doppler effect; instead, they focus on the main Doppler effect for target detection without identification. Consequently, micro-Doppler measurements are primarily conducted using prototype radars, which are still in the research phase and not yet in industrial production. Thirdly, certain targets such as birds are only encountered opportunistically and do not actively cooperate.

The generation is quite basic : the model does not account for multipath phenomena and lacks a complete representation of the drone's components. We only generated blades with a constant speed. In order to move, drones rely on a rapid change in the rotational speed of their motors to create a torque on the drone body. For future use of this dataset, it would be relevant to generate also samples that would represent a drone with a complex trajectory and a lot of variations. Here is an example of what such a difference in rotation speed would look like :

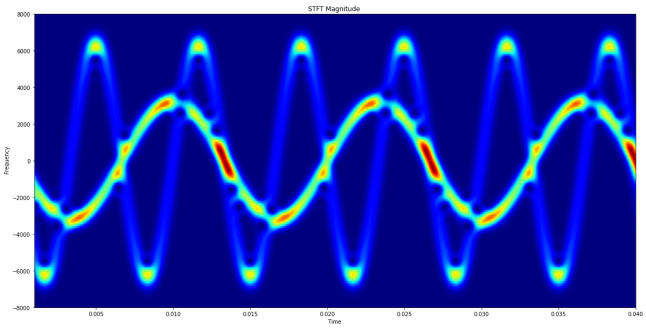


Fig. 5. Two rotors with different rotational speeds's Micro-Doppler signature (7)

To evaluate the robustness and the accuracy, it would be relevant to compare this dataset to real radar data (5)(6), especially on the radar's range and the difference in Hz between the Doppler effect of the bulk and the Micro-Doppler effect of the blades. Developing a more robust model is crucial for achieving accurate micro-Doppler signatures, where we maintain control over all parameters including radar and target variables.

Overall, the successful modeling of micro Doppler effect in radar for drone detection represents a significant advance in the field of drone detection and has the potential to greatly improve the performance and effectiveness of existing systems. This generation of datasets can for instance enable the training of both deep and non-deep Out-Of-Distribution Detection (OODD) methods (8) (9).

Bibliography

1. Jeff Sanny William Moebs, Samuel J. Ling. University physics volume 1. *OpenStax*, Sep 19, 2016.
2. Dimitrios. The doppler effect explained by comparing a static and a moving sound source. https://stock.adobe.com/fr/images/the-doppler-effect-explained-by-comparing-a-static-and-a-moving-sound-source/388859337.
3. SHEN-SHYANG HO VICTOR C. CHEN, FAYIN LI and HARRY WECHSLER. Micro-doppler effect in radar: Phenomenon, model, and simulation study. *IEEE*, 2006.
4. MathWorks. Introduction to micro-doppler effects. https://fr.mathworks.com/help/radar/ug/introduction-to-micro-doppler-effects.html.
5. Deren Li Jiangkun Gong, Jun Yan and Deyong Kong. Detection of Micro-Doppler Signals of Drones Using Radar Systems with Different Radar Dwell Times. *MDPI*, September, 2022.
6. Samiur Rahman Duncan A. Robertson. Radar micro-Doppler signatures of drones and birds at K-band and W-band. *Scientific Reports*, 2018.
7. Thomas Albert. Micro-Doppler modeling for AUV's. *ENSTA Bretagne*, February, 2024.
8. Martin Bauw, Santiago Velasco-Forero, Jesus Angulo, Claude Adnet, and Olivier Airiau. Near out-of-distribution detection for low-resolution radar micro-doppler signatures. *arXiv preprint arXiv:2205.07869*, 2022.
9. Julien Gérard. *Drone Recognition by Deep Learning*. PhD thesis, Université Paris-Saclay, 2022.

Feasibility Study of Shape Control for an Immersed Cable

Bastian Garagnon[1]

[1]Lab-STICC, ENSTA Bretagne, France

March 1, 2024

Abstract

In the context of exploring a karstic environment, the use of a cable for energy and information transfer between the robot and the surface can be a crucial asset in reducing risks and enhancing results. However, the cable should not become the weak point of the system, posing additional challenges. To prevent the cable from getting stuck on the walls, the ability to control its shape could be a viable solution. Furthermore, if the cable gains autonomy, it might be possible to unwind it from the outside, relieving the robot of this task. This could extend the reach of a teleoperated mission by eliminating the constraint of accommodating a long coiled cable within the robot. The energy and computational requirements for controlling a cable of several hundred meters are likely to be significant but can be supplied by the surface where computing and power resources are abundant and cost-effective. The cable specifications include the capability to pass a 230V AC power cable to supply the cable's actuators and the robot, an optical fiber for data transmission, and the necessary hardware to capture the cable's shape for feedback control. The aim of the study is to provide initial insights for evaluating the feasibility.

1 Introduction

Karstic exploration stands out as one of the most demanding and captivating missions in underground exploration. Furthermore, the stakes are significant due to the substantial quantity of freshwater contained in Karsts in France, as depicted in Figure 1. To contribute to the success of these missions, an immersed and actively controlled power cable for supplying an exploration robot could prove to be an excellent asset. This article aims to clearly define the objectives of the immersed power cable, the extreme constraints associated with karstic exploration missions, and discuss conceivable technical solutions and their respective limitations.

1.1 Objectives

Position Maintenance and Robot Tracking: The cable, designed to follow the exploration robot throughout its mission, must be capable of adjusting its shape along its entire length. Swift responsiveness will enable self-propulsion of the cable through oscillation, thus compensating for external forces along the cable. Controlling the cable under strong currents and over an extended deployment length is a significant challenge in the context of karstic exploration. The precision of cable control is crucial to ensure the proper functioning of the exploration robot and the mission's success.
Data Transfer: Incorporate bidirectional data transfer capabilities to ensure constant communication between the exploration robot and the control center. This facilitates real-time monitoring and decision-making.

Energy Transfer: Incorporate the ability to provide high power to the exploration robot for an energy-intensive, long-duration mission.

Length and Range: Achieve sufficient lengths to allow the robotic explorer to cover a significant distance from the cave entrance. Ensure optimal range for comprehensive exploration of karstic areas, minimizing

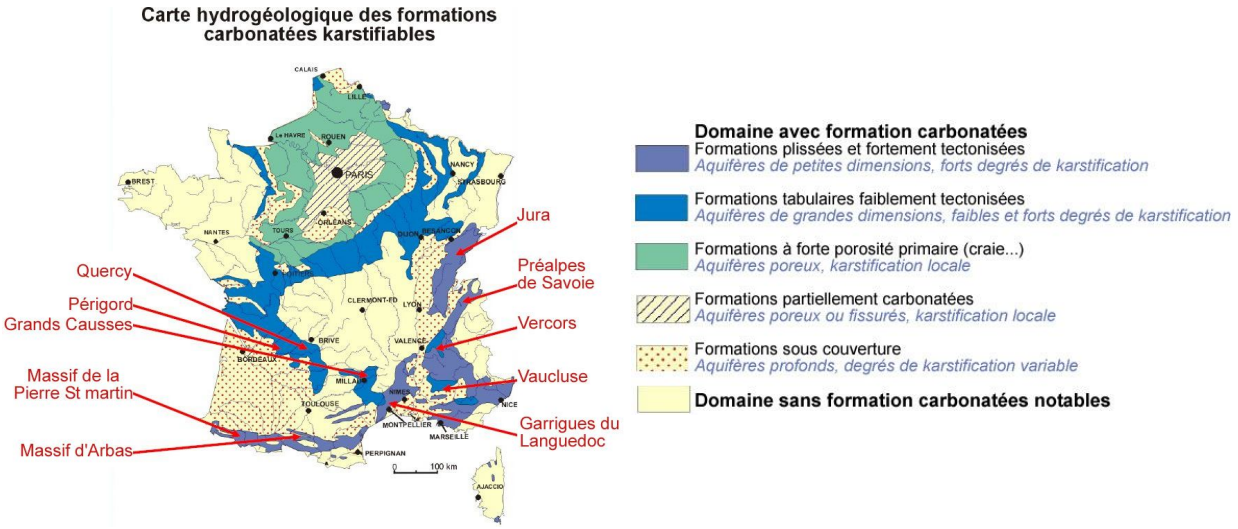


Figure 1: Presence of Karst in France [1]

geographical limitations of the mission.

Flexibility and Mobility: Allow maximum flexibility to adapt to the complex and often irregular environments of karstic caves. Facilitate the mobility of the exploration robot by providing electrical power without hindering its movements in narrow and winding passages.

Sealing and Pressure Resistance: Ensure perfect sealing to protect electrical components from water infiltration. Resist the pressure of submerged environments encountered in karstic formations, ensuring the cable's reliability at depth.

1.2 Constraints

Narrow Passages and Constrictions: Karstic caves may feature narrow passages and constrictions, necessitating a cable that is sufficiently thin and flexible to navigate through these areas without snagging or getting stuck.

Abrasion Resistance: Rock formations in karstic caves can be rough and abrasive. The cable must be designed to resist abrasion, minimizing the risks of premature wear or damage caused by friction against cave walls.

Mechanical Constraints: Twisting passages and uneven surfaces can subject the cable to significant mechanical stresses. It must be capable of withstanding torsion, bending, and other constraints without compromising its structural integrity.

Pressure Resistance: During underwater exploration in flooded karstic zones, the cable is subjected to variable pressures based on depth. It must be capable of resisting these pressures without compromising its integrity or functionality.

Lightness and Ease of Transport: To facilitate the transport of the cable to the exploration area, it is essential for it to be lightweight and easy to deploy and rewind. The size of the reel must remain reasonable despite the potentially significant cable length.

2 Specifications

Let's provide some figures to have a more precise idea of the orders of magnitude for each necessary element.

Power supplied to the exploration robot	1000W	Maximal Torque	100Nm
Power Cable Diameter	2x 3mm^2	Cable Diameter	20cm
Optical Fiber	3mm^2	Operating Depth	100m
Linear Flexibility	180°/m	Exploration Range	1000m
Swimming Speed	1m/s	Torsion	Negligible
Linear Rotation Speed	180°/s/m	Longitudinal Deformation	Negligible
Control Segment Length	10cm		

3 Actuating the Cable

Actuating the cable to a fixed length and preventing torsion requires controlling the curvature in two directions of the cable. To achieve this, we will discuss two technical solutions, the tendinous and hydraulic methods, to assess their feasibility.

3.1 Tendinous Method

In the field of karstic exploration, where precision is imperative, the use of tendinous cables (Figure 2) may appear as an innovative solution. This system, based on the use of tendons for force transmission and control of cable curvature, offers significant advantages but also presents technological challenges that may prove too complex to overcome.

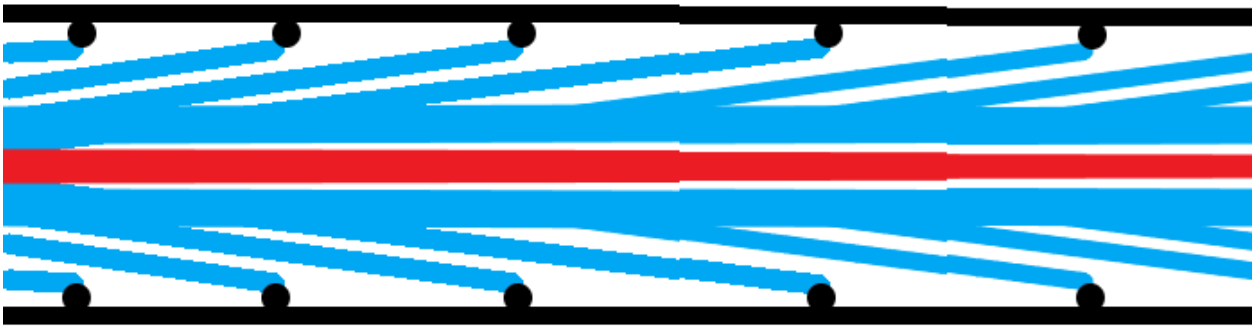


Figure 2: Tendinous cable concept
Red represents the cable's payload and the part that stiffens the cable in torsion and compression. Blue represents the tendons, and black represents the flexible sheath in which the entire structure is encased.

3.1.1 Advantages of Tendinous Cables in Karstic Exploration

Precision in Command: Tendinous cables provide exceptional precision in command transmission, with direct and responsive control. This enables fine and precise control of the cable curvature during exploration.

Positioning of Actuators Outside the Karstic Environment: By placing actuators outside the karstic environment, constraints related to compactness and mechanical resistance of internal components are signifi-

cantly reduced. This approach also allows more accessible maintenance and increased durability of actuators. The cost of actuators is also reduced as they do not require waterproofing.

Estimation of Cable State through Actuator Measurements: Sensors positioned on the actuators provide valuable information about the cable's state. Knowing the extensions of each tendon allows for an approximate reconstruction of the cable's shape. This enables continuous and real-time monitoring of the robot's position during exploration, enhancing operational reliability.

Optimal Reactivity: Tendinous cables offer optimal reactivity through direct force transmission. This results in fast response times, adaptability to sudden changes in the environment, and increased speed as long as the forces do not exceed the tendon's resistance.

3.1.2 Disadvantages and Technological Challenges

Linear Charge Losses due to Tendon Friction: Linear charge losses can occur due to tendon friction within their sheath. This may lead to a decrease in the system's energy efficiency and requires design solutions to minimize these losses.

Elasticity of Tendons: The natural elasticity of tendons can introduce variations in force transmission, affecting movement precision. Adjustment and compensation mechanisms must be integrated to minimize this effect.

Robustness of Tendons: Tendon robustness is crucial in environments where adverse conditions, such as rough surfaces, may be encountered. Materials resistant to abrasion and tearing are essential to ensure system durability.

Actuation Limited to Tendons Inside the Curvature: Only tendons located inside the curvature can effectively act, potentially limiting the range of movements for the exploration robot. This requires a tailored design to maximize flexibility and the range of possible actions.

3.1.3 Numerical Application and Material

Tendinous cables could become a promising solution for karstic exploration, offering exceptional command precision and optimal reactivity. However, technological challenges related to charge losses, tendon elasticity, and robustness require the use of advanced materials and manufacturing processes. To provide a numerical example of the technical challenge, let's consider the tension in a tendon under the most disadvantageous conditions. If a tendon needs to apply a torque of 10Nm at its section, it must exert a force of 1000N at its end. Charge losses over the hundreds of meters between the actuator and the tendon's end further exacerbate this tension. A cable resistant to these forces would have a diameter of 1.5mm and an elongation of approximately +2% at this force if it were made of ultra-high-molecular-weight polyethylene (UHMWPE), the best-known consumer technology for this purpose. A tendinous cable will be constrained to have a section of variable size depending on the cable length, as we will see in the section on compactness.

3.2 Hydraulic Method

In the search for technological solutions for karstic exploration, the hydraulic cable is an intriguing option, offering significant advantages compared to the tendinous method despite some challenges. This system, based on force transmission through hydraulics, presents unique characteristics that make it attractive for karstic exploration missions. The cable concept (Figure 4) includes a passage for the payload and the shape estimation system in red, assuming that this part bears the torsion and compression/traction forces of the cable.

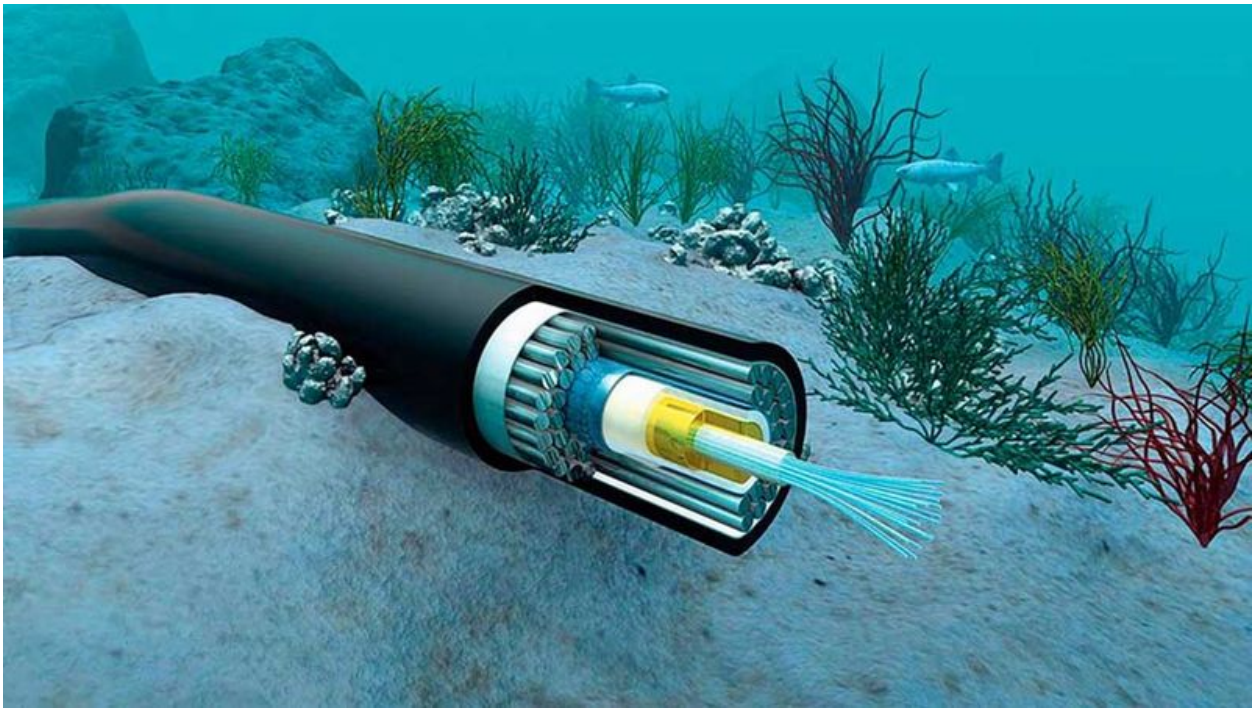


Figure 3: Conceptual 3D cut of tendinous cable

Two large hydraulic pumps outside the karst put a water tube in dark blue under high pressure and another in light blue under low pressure. Thus, small solenoid valves can supply control cells on the cable's envelope at pressures ranging between low and high pressure to control the cable's curvatures.

3.2.1 Advantages of Hydraulic Cable in Karstic Exploration

High Forces: Hydraulic cables are known for their ability to generate significant forces. This characteristic makes them particularly suitable for karstic environments, where power and force are crucial to compensate for buoyancy variations and currents, ensuring efficient swimming and high control torque.

Low Charge Losses: Hydraulic cables exhibit very low charge losses over long distances since the flow rate is relatively moderate. Therefore, cable length becomes less problematic for force transmission compared to the tendinous method. This allows the system to maintain optimal performance even over large distances, facilitating exploration deeper into karstic areas.

Constant Section: Unlike the tendinous method, the section of the hydraulic cable is independent of the cable length. This simplifies the system design and management, providing increased predictability during cable deployment and retraction.

3.2.2 Disadvantages and Technological Constraints

Cable Pressure Resistance: Pressure resistance is a major challenge for hydraulic cables, especially when used in karstic environments where pressures can vary with depth. Specific materials and structures must be implemented to ensure the cable's resistance to both internal and external pressure variations.

Rigidity: Hydraulic cables operate under high pressures to generate the necessary force. This can result in some cable rigidity, limiting its flexibility and ability to follow complex contours in intricate underground environments. This may necessitate the use of even higher pressures to increase control torque, further

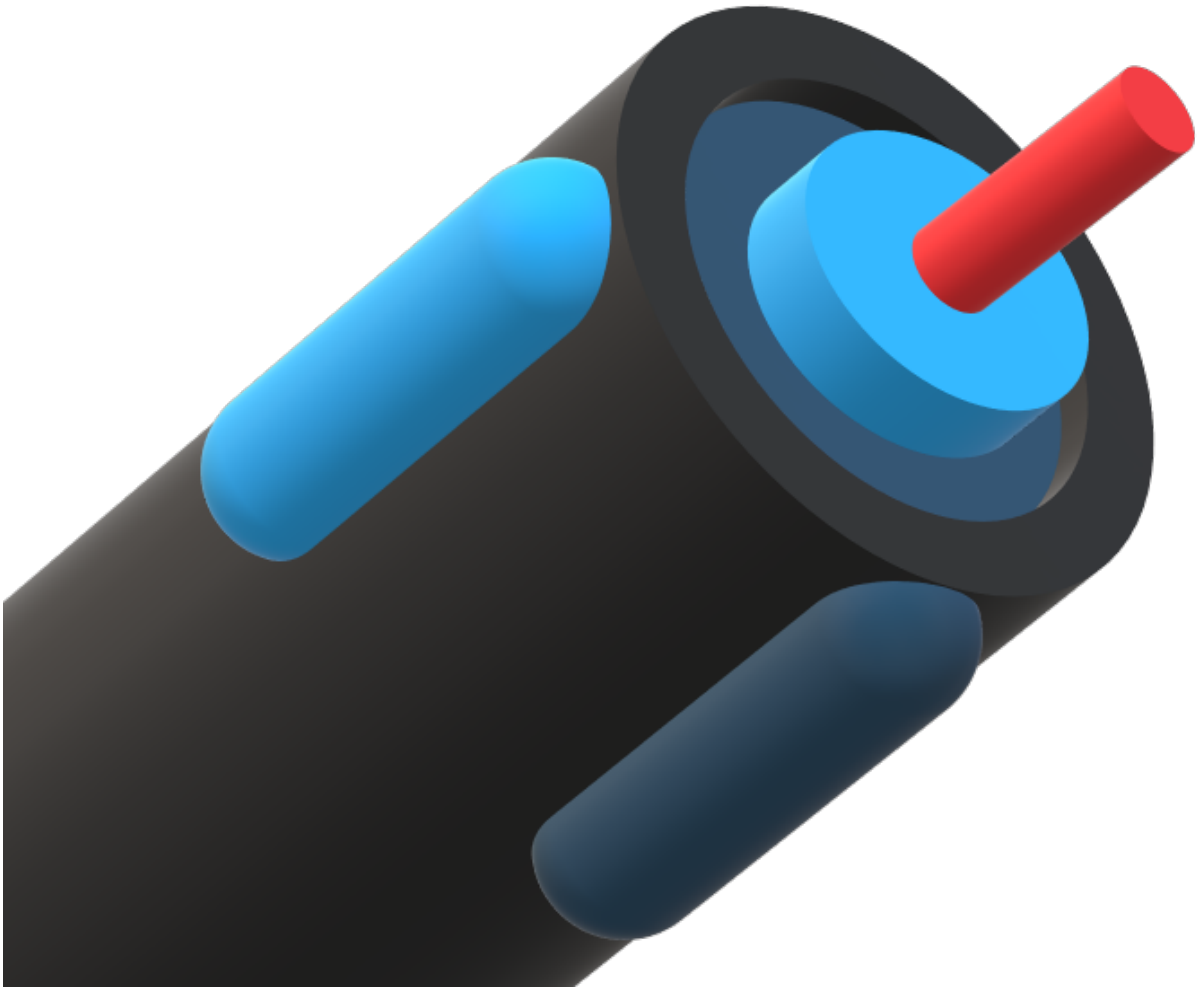


Figure 4: Conceptual 3D cut of hydraulic cable

complicating the design.

Presence of Actuators: Integrating solenoid valves and pressure control mechanisms into the cable can pose design challenges. This may increase the system's complexity and require advanced technologies to ensure precise and responsive control. In particular, establishing reliable communication between the exterior and solenoid valves requires a dedicated network that needs to be both reliable and integrated into the cable.

3.2.3 Numerical Application and Material

The hydraulic cable offers an exciting prospect for karstic exploration by providing significant forces and a constant section over long distances. However, challenges related to pressure resistance, stiffness due to high pressure, and the integration of actuators require addressing numerous technical hurdles. To provide a numerical example of the technical challenge, we can elaborate on the pressures needed under the most disadvantageous conditions. Two cells of a $13cm^2$ section (2cm in diameter), positioned opposite each other around the cable perimeter, must generate a torque of 100Nm. Thus, the pressure difference must be 8 bars if the cable's rigidity is zero. Therefore, a low-pressure difference around 1 bar and a high pressure around 20 bars will be considered to ensure the ability to meet the specifications.

4 Study of Bulkiness

Bulkiness is an important parameter to ensure that the cable can navigate through narrow karstic passages and for transportation purposes. We will discuss the bulkiness of both methods to characterize them in terms of cable diameter.

4.1 Cable Diameter

For the tendinous method, the diameter of one tendon is 1.5mm, and as four tendons are the minimum for optimal transmission of efforts for both bends, each section requires 9mm^2 of tendons in all previous sections. The last section, closest to the exit, will have a minimum diameter of 15cm if we consider a section every 10 cm and a cable length of 1000m. The tendinous cable roughly meets the specifications. However, the diameter of the last section must increase at $O(\sqrt{L})$, where L is the cable length, to provide a sufficiently large section for the passage of tendons.

The hydraulic method, on the other hand, can simply adapt to the 20cm diameter regardless of the cable length. The medical industry already uses miniature solenoid valves for artificial hearts, for example, which measure only a few tens of millimeters in length. Their low energy consumption also limits the bulkiness of the power and control network of the solenoid valves. The 1000m cable will be filled with a water volume of around 80m^3. If the entire cable oscillates to swim at 1m/s (the most disadvantageous situation), a small part of the total volume will need to be renewed every second, approximately one-eighth. Thus, the flow rate will be about $10\text{m}^3/\text{s}$ (which is huge). In conduits as described in the concept (Figure 4), this corresponds to an 80% loss of charge after 1000m. Thus, the beginning of the cable would only benefit from a sufficient pressure difference, which would be problematic for maintaining control torque. Therefore, to limit the flow rate that the pump must provide and the losses of charge, it will be necessary to limit the part of the cable that swims by oscillating at full speed to 10%.

5 Conclusion

In conclusion, the use of controlled cables for karstic exploration presents real potential, paving the way for possibilities such as securing exploration, the ability to carry more power-demanding equipment, and real-time processing of data from the exploration robot. However, these potential advantages come with considerable technical challenges that require a thoughtful and innovative approach.

For tendinous cables, the complexity of their fabrication and the strength of tendons are major challenges to overcome. Hydraulic cables, on the other hand, must address issues related to high flow rates and pressures, as well as manufacturing challenges. Additionally, accurately detecting the shape of the cable over long distances remains a complex challenge, although advancements can be envisioned as technology evolves.

Another crucial challenge lies in the development of control algorithms tailored to these cables. Their spatial extent means that each part of the cable must be autonomous in managing currents and other constraints while being interconnected with other parts to ensure overall and consistent adaptation.

Although these challenges are significant, the potential benefits of using controlled cables in karstic exploration could revolutionize our ability to explore these complex and mysterious environments. Ongoing innovation in the fields of fabrication, shape detection, and control algorithms can pave the way for innovative technological solutions and lead to significant advancements in the field of karstic exploration.

6 Bibliography

[1] Bakalowicz, M., & Kalfoun, F. (2003). Karst et érosion karstique. Hydrosciences, UMR 5569, Montpellier, France.

[2] Kim, J., Seo, Y., Choi, S., & Kim, B. G. (2023). Distal End Force Estimation of Tendon-sheath Mechanism Using a Spring Sheath. International Journal of Precision Engineering and Manufacturing, 24(12), 1-13. DOI: 10.1007/s12541-023-00917-1.

[3] Lee, D. G., Baek, D., Baek, D., Kim, H., Kim, H., & Kwon, D. S. (2022). Learning-Based Discrete Hysteresis Classifier Using Wire Tension and Compensator for Flexible Endoscopic Surgery Robots. International Journal of Precision Engineering and Manufacturing, 24(2). DOI: 10.1007/s12541-022-00716-0.

[4] Almaghout, K., & Klimchik, A. (2024). Manipulation Planning for Cable Shape Control. Robotics, 13(1), 18. DOI: 10.3390/robotics13010018.

[5] Tortorici, O., Peraud, C., Anthierens, C., Hugel, V. (2024). Automated Deployment of an Underwater Tether Equipped with a Compliant Buoy–Ballast System for Remotely Operated Vehicle Intervention. Journal of Marine Science and Engineering, 12(2), 279. DOI: 10.3390/jmse12020279.

Resolving the catenary equation for a ROV evolving in "hook" configuration

Guillaume Garde
ENSTA Bretagne, Robotique Autonome, FISE 24

March 3, 2024

Abstract

This paper presents the modeling of the umbilical of a Remotely Operated Vehicle (ROV) evolving in a special type of configuration called the "hook" configuration. The model presented here is based on the resolution of the catenary equation in the case of an underwater cable whose ends float at distinct depths. The approach is based on the variational principle used to try and minimize the potential energy of the cable. The form of the solution to this problem is given and the coefficients are estimated numerically. The static shape of an underwater cable fixed to a surface vessel and to an underwater floater is computed accordingly and presented.

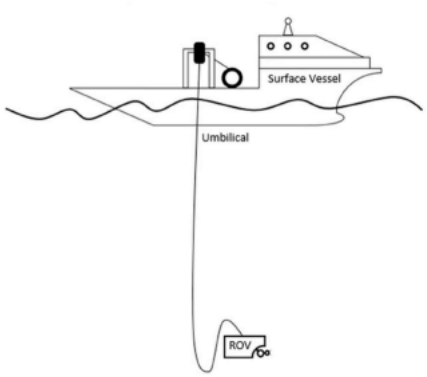


Figure 1: The "hook" configuration [5]

1 Introduction

The management of underwater ROV umbilical cables is a well-known problem. With the development of deep-water exploration, it has become necessary to understand the dynamics of said cables to safely deploy or maneuver ROVs. The combination of static and dynamic tension reveals to be raising many issues, such as snap loading [4][6] or self winding [5]. Hence the need to correctly model underwater cable dynamics and to find methods to safely manage them. Lubis et al. propose and compare in [5] four cable configurations for deep water exploration. The novelty brought by [5] is the proposal of a "hook" configuration for minimizing tension. The visual description of such a configuration is shown in figure 1. The paper also introduces three other variations of this configuration by adding floaters to the cable. The "mid-depth hook" configuration, for instance, uses floaters in the approximate middle of the umbilical as shown in figure 2. This configuration is supposed to be better for more severe environments. It decouples the ROV motion from the surface vessel motion and reduces the static tension in the top part of the cable. The idea is that a zero-tension zone appears around the floaters. This configuration motivates this paper by raising the question of a simple model for the form of the cable between the surface vessel and the top floater. The approaches to modeling underwater cables are varied and include finite elements [2] or finite difference [1] approaches. This paper proposes another approach based on the variational principle. It is sequenced as follows: First, a quick statement of the problem before moving on to the mathematical reasoning and finally some numerical approximations to estimate the parameters of the model.

2 Problem statement

Let $R(0, x, y)$ be the Galilean work frame of this study. The aim of this section is to model the static behavior of an underwater cable connecting a surface vessel at point (x_0, y_0) to the top floater at point (x_f, y_f) according to the "mid-depth hook" configuration [5]. Let l be the length of the cable, ρ its density, S its section, and g the gravitational constant. The cable is considered unstretchable and of uniform density.

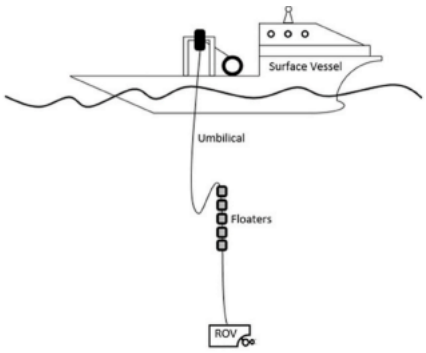


Figure 2: The "mid-depth hook" configuration [5]

3 Variational principle-based reasoning

The *principal of least action* [3] argues that the static shape of the cable minimizes its potential energy. Therefore, the problem can be reduced to the research of the shape $y \mapsto y(x)$ that minimizes the functional E defined by:

$$\begin{aligned} E[y] &= \int_{cable} \rho g S y ds \\ &= \int \rho g S y \sqrt{(dx^2 + dy^2)} \\ &= \int_{x_0}^{x_f} \rho g S y \sqrt{1 + \dot{y}^2} dx \end{aligned} \tag{1}$$

with the length constraint[1]:

$$l = \int_{x_0}^{x_f} \sqrt{1 + \dot{y}^2} dx \tag{2}$$

Consequently, the simultaneous consideration of (1) and (2) leads to the minimization of the functional $\hat{E}$ such that:

$$\hat{E}[y] = \int_{x_0}^{x_f} h(y, \dot{y}) dx \tag{3}$$

where

$$h(y, \dot{y}) = \rho.g.S.y.sqrt(1 + \dot{y}^2) + \lambda.sqrt(1 + \dot{y}^2) dx \tag{4}$$

In (4), λ is a Lagrange multiplier whose aim is to take the length constraint into account in the calculations. The function h minimizing (3) verifies the *Euler-Lagrange* equation [3]:

$$\frac{\partial h}{\partial y} - \frac{d}{dx}(\frac{\partial h}{\partial \dot{y}}) = 0 \tag{5}$$

Furthermore, h does not explicitly depend on x and therefore verifies the *Beltrami* formula:

$$h - \dot{y} \frac{\partial h}{\partial \dot{y}} = C \tag{6}$$

[1]The cable is unstretchable.

where C is a constant to be dertimined. The problem now consists of solving (6). A little manipulation of the equation gives that:

$$\sqrt{1 + \dot{y}^2} = \frac{\rho g S y + \lambda}{C} \tag{7}$$

that is:

$$1 + \dot{y}^2 = (\frac{\rho g S y + \lambda}{C})^2 \tag{8}$$

Let then z be equal to $\frac{\rho g S y + \lambda}{C}$. Then (8) gives that:

$$1 + (\frac{C}{\rho g S})^2 \dot{z}^2 = z^2 \tag{9}$$

which means that:

$$\dot{z}^2 = (z^2 - 1)(\frac{\rho g S}{C})^2 \tag{10}$$

Appears the differential equation:

$$\begin{aligned} \dot{z} &= \sqrt{(z^2 - 1)(\frac{\rho g S}{C})} \\ \dot{z} &= -\sqrt{(z^2 - 1)(\frac{\rho g S}{C})} \end{aligned} \tag{11}$$

One solution to (11) has the form $z \mapsto z(x)$

$$z(x) = \cosh(\frac{\rho g S}{C} x + B) \tag{12}$$

where B is to be determined. Going back to y, (12) means that:

$$y(x) = \frac{1}{\rho g S}[C \cosh(\frac{\rho g S}{C} x + B) - \lambda] \tag{13}$$

Here comes the fact that $y(x_0) = y_0$ and that $y(x_f) = y_f$. Developing these limit conditions gives two new equations:

$$\begin{cases} \lambda = C \cosh(\frac{\rho g S}{C} x_0 + B) - \rho g S y_0 \\ \lambda = C \cosh(\frac{\rho g S}{C} x_f + B) - \rho g S y_f \end{cases} \tag{14}$$

Finally, using (13) to develop (2), the length constraint, gives that:

$$l = \frac{C}{\rho g S}[\sinh(\frac{\rho g S}{C} x_f + B]) - \sinh(\frac{\rho g S}{C} x_{x0} + B)] \tag{15}$$

With (14) and (15), the number of useful equations to find missing coefficients is 3, just as much as the undetermined values λ, C and B.

4 Numerical finding of the coefficients

λ, C and B must be found in order to fully solve the problem, but it would be too difficult to solve the non linear system of equations given by (14) and (15) "by hand". The idea here is to rely on numerical methods to approximate the values. Three different methods are presented here, all based on *Python* language and librairies.

Method	Number of iterations	Calculation time in seconds	Results
scipy and fsolve	10000	0.7984063625335693	$C = 5.98188473$ $B = -2.13404675$ $\lambda = 25.62443807$
scipy and least_squares	10000	2.28048038482666	$C = 5.98188473$ $B = -2.13404675$ $\lambda = 25.62443807$
sympy and solve	1	Not finished (over 2 hours)	None

Figure 3: Numerical results

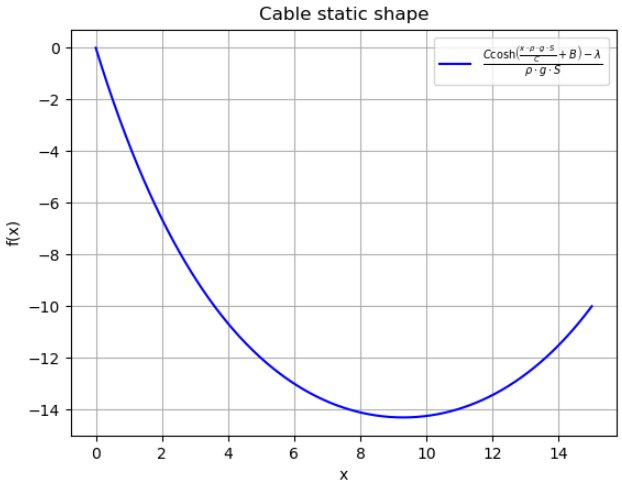


Figure 4: Static shape of the cable with numerically estimated parameters.

1. using *scipy.optimize* and the *fsolve* method
2. using *scipy.optimize* and the *least_squares* method
3. using *sympy* and the *solve* method

The figure 3 summarizes the results. The calculations have been made for a cable of length 30 meters and of section 1 squared centimeters, attached at point $(0, 0)$ and point $(15, -10)$ (in meters), and of density 1,4. The curve corresponding to the solution is given in figure 4.

5 Conclusion

The resolution of the catenary equation for an underwater static cable has been successful, and numerical methods gave parametric estimations that have allowed the reasoning to go further. The idea motivating this paper was to find the best shape for an underwater cable, minimizing its potential energy while respecting the "mid-depth hook" configuration. This work might motivate a future paper on the control of a floater to ensure that a ROV's umbilical gets as close as possible to the profile developed here.

References

[1] C.M. Ablow and S.Scheshter. Numerical simulation of undersea cable dynamics. *Ocean Engineering 10*, 1983.

[2] Ole Alexander, Nørve Eidsvik, and Ingrid Schjølberg. Finite element cable-model for remotely operated vehicles (rovs) by application of beam theory. *Ocean Engineering 163*, 2018.

[3] Richard Feynman. *The Feynman lectures on physics vol.2*. 1963.

[4] Shan Huang and Dracos Vassalos. A numerical method for predicting snap loading of marine cables. *Applied Ocean Research 15*, 1993.

[5] Michael Binsar Lubis, Mehrdad Kimiaei, and Mike Efthymiou. Alternative configurations to optimize tension in the umbilical of a work class rov performing ultra-deep-water operation. *Ocean Engineering 225*, 2021.

[6] Christophe Viel. Self-management of the umbilical of a rov for underwater exploration. *Ocean Engineering 248*, 2022.

Robust path planning for autonomous robots

Louis Gillard
ENSTA Bretagne
Brest, France
louis.gillard@ensta-bretagne.org

I. Introduction

In the realm of autonomous robotics, the ability to navigate through complex and dynamic environments is a critical factor that directly influences the overall performance and success of a robotic system. Robust path planning, the process by which autonomous robots determine an optimal path from their current location to a specified goal while avoiding obstacles and adapting to uncertainties, has become a crucial area of research. This field addresses the challenges of scenarios with dynamic obstacles and environmental changes.

A proficient path planning algorithm should meet four essential criteria. Initially, the motion planning technique must consistently identify the optimal path within realistic static environments. Additionally, the algorithm should be adaptable to dynamic environments, ensuring its effectiveness in scenarios with evolving conditions. Furthermore, it should seamlessly integrate with and augment the chosen self-referencing approach. Lastly, the algorithm must prioritize minimizing complexity, reducing data storage requirements, and optimizing computation time. These criteria collectively contribute to the algorithm's efficiency, versatility, and compatibility with various real-world applications. In order to comply with these restrictions, we must come up with algorithms tailored to the unique characteristics of each environment. Consequently, the focus of this document revolves around addressing the following question: What are the current state-of-the-art path-planning algorithms in use today ?

To organize the exploration of path-planning algorithms effectively, we categorize them into three distinct sections: Traditional algorithms, Graph search algorithms and Group optimization algorithms.

II. Traditional algorithms

A. Artificial potential field algorithms

The fundamental concept behind the Artificial Potential Field (APF) algorithm [1] involves creating a simulated force field. In this framework, autonomous vehicles are represented as mass points, and environmental data is portrayed through a combination of attractive forces from the target gravitational field and repulsive forces from obstacles. Figure 1 gives an example of such algorithm. The farthest the vehicle to the goal, the more attracted it gets, while a repulsive force emerges from the central obstacle. By guiding the mass points along the direction of the resultant force, the APF algorithm achieves the generation of a seamless, collision-free path.

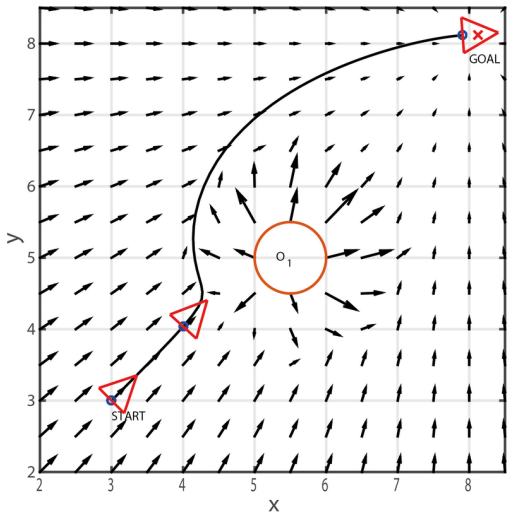


Fig. 1. A visualization of an artificial potential path planning algorithm [2]

B. Djikstra algorithms

The Dijkstra algorithm operates by addressing subproblems, computing the shortest path from the source to vertices among the nearest vertices to the source [3]. It identifies the subsequent closest vertex by managing new vertices in a priority-min queue and retains only one intermediate node, limiting the determination of a single shortest path.

Dijkstra efficiently calculates the shortest path from the starting point to every point. Various adaptations of the Improved Dijkstra's algorithm have emerged, each tailored to specific applications, showcasing the algorithm's versatility across diverse use cases.

While the traditional Dijkstra algorithm relies on a greedy approach for path planning, primarily focused on finding the shortest path in a graph, it lacks consideration for the practicality of the solution. In response, we can see modified Dijkstra's algorithms that introduce a novel component by incorporating probabilities along each edge of the graph [4]. This modification aims to discover alternative routes, particularly useful when the costs associated with generating plausible shortest paths are substantial. The introduction of probabilities enhances the algorithm's flexibility, addressing computational limitations and expanding its applicability to novel scenarios.

C. Rapidly-exploring random tree algorithm

The RRT algorithm is employed for decision-making in path planning as well [5]. The approach involves initiating a tree with the starting point as the root node and then randomly expanding the tree, with a given constant step length, within the feasible space until it reaches the endpoint (leaves). Consequently, a collision-free path is derived from the initial to the final point. This data structure is hugely useful for handling path-planning queries with non-holonomic constraints involving dynamics and high degrees of freedom. Figure 2 shows the result of such algorithm without obstacle. We can see how the whole space is quickly covered up by the nodes.

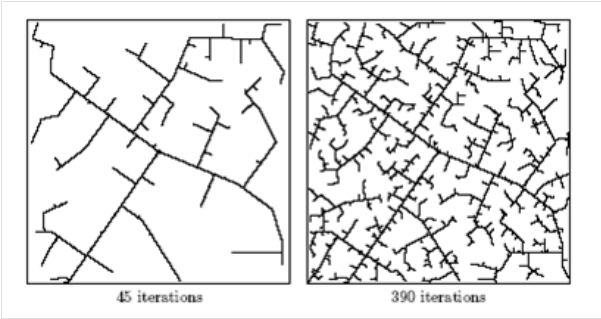


Fig. 2. A visualization of an RRT graph after 45 and 390 iterations [6]

D. Simulated annealing algorithms

Simulated Annealing (SA) is a probabilistic optimization algorithm inspired by the annealing process in metallurgy. It is often used for solving optimization problems, including path planning [7].

E. Tabu search algorithms

Tabu Search is a metaheuristic algorithm used for solving optimization problems. It is particularly effective in navigating large solution spaces where traditional search algorithms may struggle to find optimal solutions [7].

III. Graph search algorithms

Graph search algorithms operate by representing a map through a grid method, wherein the map is decomposed into interconnected, non-coincident grids. The objective is to search for an optimal path from the starting grid to the target grid, effectively avoiding collisions in the process. Notably, the A* and D* algorithms have gained popularity in the field of path planning for autonomous vehicles.

The A* algorithm [8] is a hybrid approach that integrates aspects of both the Dijkstra algorithm and the Best-First Search algorithm. This algorithm establishes an open list and a closed list. The open list contains grid points available for selection, while the closed list holds the grid points that form the selected path. The process begins by placing the starting grid number of the autonomous vehicle in the open list. Subsequently, adjacent grids that the vehicle may pass through are added to the open list. The evaluation function, denoted as f(n), is computed for the adjacent grid of the starting point in the open list. The starting point is then moved to the closed list, selecting the grid point with the smallest value as the new starting point. This loop continues until the target point raster is placed in the open list. Finally, the points in the closed list are sequentially connected to derive the optimal path. The figure 3 gives a broad idea of the algorithm. Each grid cell being a node whose value in the algorithm is expressed. The valuation function of the algorithm is generally expressed as:

$$f(n) = g(n) + h(n) \tag{1}$$

where g(n) represents the actual cost from the starting point to the current point, and the heuristic function h(n) estimates the cost from the current point to the target point. Typically, the Euclidean distance serves as the metric for the cost function. Altering the cost function can significantly enhance the performance of the A* algorithm. The following table shows a few very common heuristic functions.

TABLE I
Most Common Heuristic Functions

Heuristic	Expression
Euclidean distance	$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$
Manhattan distance	$\lvert x_1 - x_2\rvert + \lvert y_1 - y_2\rvert$
Octile distance	$\max(\lvert x_1 - x_2\rvert, \lvert y_1 - y_2\rvert)$

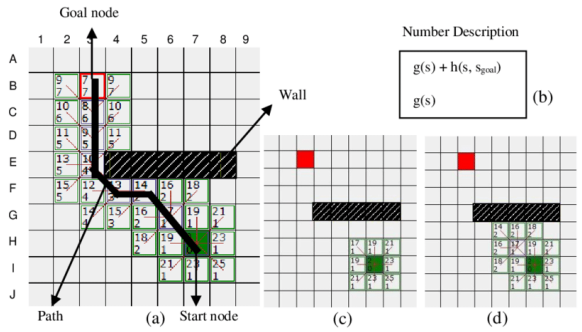


Fig. 3. A visualization of the A* algorithm [9]

IV. Group optimization algorithms

The group intelligent optimization algorithm emulates the cluster behavior observed in biological groups, where each member continuously refines its search direction based on personal experiences and learns from the collective experience of other group members. In the domain of path planning for autonomous vehicles, two prominent algorithms, namely the ant colony algorithm (ACO) [10] and particle swarm optimization (PSO) [11] algorithm, leverage the principles of group intelligence.

The ACO algorithm draws inspiration from the foraging behavior of ant colonies. Individual ants conduct random

searches, and each agent determines its next step by referencing the pheromones left by fellow ants. However, it's essential to note that pheromones gradually dissipate over time.

To implement path planning for autonomous vehicles using the ACO algorithm, the following procedure is adopted. Initially, all ants are positioned at the vehicle's starting point, and each ant selects its subsequent position based on state transitions until it reaches the destination or its path becomes invalid. Once all ants have completed their journeys, the optimal path is chosen for the entire ant colony through an iterative process. The state transition formula, a pivotal element of the ACO algorithm, governs this decision-making process.

Figure 4, shows the ants converging to a path that is not optimal but shorter than the other possible paths.

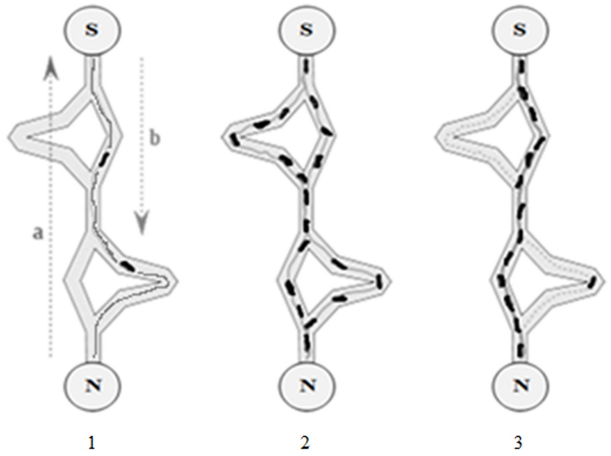


Fig. 4. A visualization of the ACO algorithm [12]

V. Conclusion

In conclusion, each algorithm discussed in this paper presents a unique set of advantages and limitations, emphasizing the importance of selecting the most suitable approach based on specific application requirements and environmental considerations. The diverse range of algorithms covered underscores their widespread adoption and adaptability across various domains in autonomous robotics. The comparison of these algorithms, depicted in Figure 5, provides valuable insights into their relative strengths, challenges, efficiency, and stability.

Category	Name	Advantages	Problems	Efficiency	Stability
Traditional algorithm	APF[10]	Simple structure	Easy to fall into local optimum	High	Low
	RRT[11]	Simple structure	Narrow space search failure	Middle	Middle
	SA[7]	Simple structure	Sensitive to parameter selection	Low	High
	TS[8]	Simple structure	Easy to fall into local optimum	Low	Middle
Graph search algorithm	A*[29]	Complicated structure	Only suitable for static path planning	High	High
	D*[16]	Complicated structure	Only for short distance dynamic path planning	High	High
Group intelligent optimization algorithm	ACO[20]	Complicated structure	Easy to fall into local optimum	Low	High
	PSO[21]	Simple structure	Easy to fall into local optimum	Low	High
	IWD[22]	Simple structure	Easy to fall into local optimum	Low	Middle
	GA[24]	Simple structure	Easy to fall into local optimum	Low	Middle
	ANN[26]	Complicated structure	Data processing is opaque	Low	High
	Q-learning[28]	Simple structure	Long learning time	Low	High

Fig. 5. Comparison of some algorithms described in the present document [13]

References

[1] Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. In: Autonomous robot vehicles, pp. 396–404 (1986)

[2] 'Traditional artificial potentials path planning', ResearchGate. Available at: https://www.researchgate.net/figure/Traditional-artificial-potentials-path-planning_fig1_320174864

[3] Dijkstra, E. A note on two problems in connexion with graphs. Numer. Math. 1959, 1, 269–271. [CrossRef]

[4] Gbadamosi, O.A.; Aremu, D.R. Design of a Modified Dijkstra's Algorithm for finding alternate routes for shortest-path problems with huge costs. In Proceedings of the 2020 International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS), Lagos, Nigeria, 18–21 March 2020; pp. 1–6. [CrossRef]

[5] Lavalle, S.: Rapidly-exploring random trees: A new tool for path planning. Technical Report, pp.98–11 (1998).

[6] 'Rapidly exploring random tree', Wikipedia. Mar. 01, 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Rapidly_exploring_random_tree

[7] Ming, Yu, Yanqiang Li, Zihui Zhang, et Weiqi Yan. 'A Survey of Path Planning Algorithms for Autonomous Vehicles'. SAE International Journal of Commercial Vehicles 14, no 1 (24 janvier 2021): 02-14-01-0007. https://doi.org/10.4271/02-14-01-0007.

[8] Hart, E., Nilsson, J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics, 4(2): 100–107 (1968)

[9] 'Fig. 2. Illustration of A* algorithm path planning', ResearchGate. [Online]. Available: https://www.researchgate.net/figure/Illustration-of-A-algorithm-path-planning_fig1_283508751

[10] Dorigo, M., Maniezzo, V., Colorni, A. : Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 26(1): 29–41 (1996)

[11] Shi, Y.: Particle swarm optimization: developments, applications and resources. Proceedings of the Congress on evolutionary computation, pp. 81–86 (2001)

[12] 'Ant Colony Optimization Algorithm processes.', figshare. [Online]. Available: https://plos.figshare.com/articles/figure/_Ant_Colony_Optimization_Algorithm_processes_/1418788/1

[13] Y. Ming, Y. Li, Z. Zhang, and W. Yan, 'A Survey of Path Planning Algorithms for Autonomous Vehicles', SAE Int. J. Commer. Veh., vol. 14, no. 1, pp. 02-14-01–0007, Jan. 2021, doi: 10.4271/02-14-01-0007.

Learning how to bowl: a hybrid approach between reinforcement learning and classical physics.

Clara Gondot
ENSTA Bretagne
Mobile Robotics student
Brest, France

***Abstract*—This article uses a precise dynamic model describing a bowling ball trajectory to build a simulation environment suited for reinforcement learning. An agent with the mission of throwing the ball is trained in this environment with a hybrid algorithm, using both a simple physics-based model and reinforcement learning.**

***Index Terms*—reinforcement learning, bowling, robotic arm, dynamic model**

INTRODUCTION

In the realm of robotics and automation, physics-based analytical models are often used to create controllers capable of accurate dynamic motions. Their main drawback is their reliance on a precise and unchanging environment. In scenarios with uncertain conditions, machine learning algorithms tend to perform better. By learning from data and experience, these algorithms exhibit a remarkable capacity to generalize behaviors and adapt to diverse scenarios. However, their reliance on vast amounts of training data and their tendency towards approximations can compromise accuracy, especially for complex dynamic maneuvers.

This article focuses on the act of throwing. In the work of Zeng et al. in [7], the *Tossingbot* appeal is its ability to quickly organize arbitrary objects by tossing them in a delimited landing zone. The method uses an analytical model of the objects trajectory to which is added residuals computed with reinforcement learning. In a similar manner, the work presented here tries to fuse a physic-based controller with reinforcement learning. As there are numerous different kind and contexts for throwing, the application chosen was ten-pin bowling. The allure of this application, apart from entertainment, is that the dynamic model of a bowling ball along the line is actually quite complex, including because of the mass distribution in the ball. The main objective of this article is to test the method presented in [7], in a situation where the physics-based model uses important simplifications and to study the behaviour of its reinforcement learning counterpart.

In Section I, the overall context of the environment is defined, as well as the simplified model of the bowling ball used by the physics-based controller. The second section introduces the simulation environment used to test the control algorithms implemented. Some results are exposed and interpreted. Lastly, Section III presents the principles behind the Machine Learning controller and perspectives for a future implementation.

I. MODELLING

A. Environment description

In the application described in this article, an artificial agent is replacing a bowling player. As a player, the agent is placed before a bowling lane, and releases the ball at the foul line, see Figure 1.

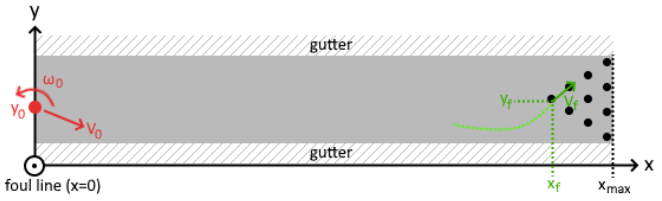


Fig. 1. Representation of the input parameters (in red) used as initial conditions in the analytic model, and the expected output or results (in green), corresponding to the moment the ball touches the headpin.

A bowling lane is usually oiled, and the oil pattern as well as the associated friction coefficients can vary. In light of Ji et al. study in [4], it is assumed that until 12,2 meters, the lane friction coefficient is of 0.04, and 0.2 after. This is a simple but sufficient model for the wanted application.

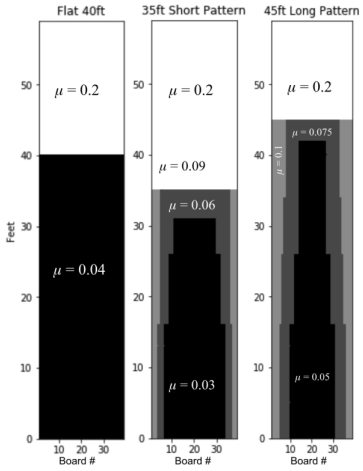


Fig. 2. Different oil patterns used in competitive bowling. The far left figure corresponds to the pattern use in this article. Figure taken from [4].

The expected inputs the agent has to decide upon, are the initial linear speed $\vec{V_0}$, supposed horizontal, angular speed $\vec{\omega_0}$,

on the contrary not necessarily horizontal, and ordinate y_0 on the foul line. This total of 6 parameters has to optimize the probability of a strike happening.

B. Simplified analytic model

In this subsection, some hypothesis made are overly simplifying the behaviour of our ball. This way, the analytical study used for the control of the agent is also simplified and the machine learning algorithm described in Section II will supposedly cover for the inaccuracies.

First, the ball is assimilated to a perfect sphere, of radius R_b and mass M. As well explained by Normani in[5], the moments of inertia of the ball differ with how it was constructed. Usually, a bowling ball contains an weight block inside, with an arbitrary shape. Here, the inertia tensor I_0 is supposed diagonal, with equal components. This hypothesis implies that the axes of the principal moments of inertia stay constant. Indeed, if R is the rotation matrix of the ball in any attitude, then:

$$RI_0R^T = R(kI_{3\times3})R^T = kRR^T$$

Since R is orthogonal, $R^T = R^{-1}$, hence

$$RI_0R^T = kI_{3\times3} = I_0 \tag{1}$$

The second simplifying hypothesis made here is that the center of mass is the center of the sphere. Applying Newtons second law to our system gives us that :

$$M\ddot{\vec{r}} = -Mg\vec{z} + \vec{F}_{con} \tag{2}$$

$$\frac{d}{dt}(I_0\vec{\omega}) = -R_b\vec{z}\times\vec{F}_{con} \tag{3}$$

where $\vec{F}_{con}$ is the force exerted by the lane on the ball, $\vec{z}$ is the upward unit vector, $\vec{r}$ is the position of the ball w.r.t the origin of the lane and $\vec{\omega}$ its angular velocity.

In this model as well as the one detailed in Section III, the force $\vec{F}_{con}$ follows Coulomb's Law of friction. Following the methodology used by Frohlich in [3], the following variable $\vec{s}$ and $\vec{\mu}$ are defined as :

$$\vec{s} = -R_b\vec{z}\times\vec{\omega} - \dot{\vec{r}} \tag{4}$$

the speed between the lane and the contact point of the ball on the lane, and

$$\vec{\mu} = \begin{pmatrix}\mu_x\\ \mu_y\\ 1\end{pmatrix}, \text{ where } \mu\frac{\vec{s}}{||\vec{s}||} = \begin{pmatrix}\mu_x\\ \mu_y\\ 0\end{pmatrix} \tag{5}$$

with μ the friction coefficient of the lane, as defined earlier.

When the ball is sliding, with the previous hypotheses, the contact force can be expressed as :

$$\vec{F}_{con} = -MgR_b\vec{z}\times\vec{\mu} \tag{6}$$

Because of Eq. (1), Eq. (3) can be rewritten as :

$$I_0\dot{\vec{\omega}} = -R_b\vec{z}\times\vec{F}_{con} \tag{7}$$

By developing Eq. (7) knowing Eq. (6), the expression of $\dot{\vec{\omega}}$ is determined :

$$\dot{\vec{\omega}} = -MR_bgI_0^{-1}\begin{pmatrix}-\mu_y\\ \mu_x\\ 0\end{pmatrix} \tag{8}$$

If the ball is rolling, then $\vec{s} = \vec{0}$. This induces that :

$$\ddot{\vec{r}} = -R_b\vec{z}\times\dot{\vec{\omega}}$$

The expression of $\vec{F}_{con}$ can then be deduced from Eq. (2) and the previous expression :

$$\vec{F}_{con} = M(g\vec{z} + -R_b\vec{z}\times\dot{\vec{\omega}}) = M\begin{pmatrix}R_b\dot{\omega_y}\\ -R_b\dot{\omega_x}\\ g\end{pmatrix} \tag{9}$$

As done previously, Eq. (7) is developed knowing Eq. (9), which results in :

$$\dot{\vec{\omega}} = MR_b^2I_0^{-1}\begin{pmatrix}\dot{\omega_x}\\ \dot{\omega_y}\\ 0\end{pmatrix} \tag{10}$$

The equations (8) and (10) are fairly simplified compared to the ones developed in Section II. However, integrating the value of $\vec{\omega}$ is not that simple, as the expression of $\vec{s}$, see (4), depends on $\vec{\omega}$ and $\vec{r}$ and influences the motion of the ball.

In the rolling case, thanks to the first hypothesis about the equal moments of inertia, Eq. (9) induces:

$$\dot{\vec{\omega}} = \vec{0} \Rightarrow \vec{\omega} = \vec{\omega_0} \tag{11}$$

However, the ball often begins by sliding. A way to handle the rolling/sliding condition and output an analytical solution is still unknown.

Ultimately, the *Physical controller* will be limited to a short numerical study detailed in the next subsection, and the help of a practised mathematician to look upon and solve the bowling ball motion equations would be needed.

C. How to strike

Without the access to neither a bowling lane nor a robot capable of throwing bowling balls, a simulation, described in Section II, is used to compute the trajectory of the throw made by our agent.

To evaluate how much was scored, the study cited by Benner et al. in [2] will be exploited as it is in [4]. The state of the ball, its velocity and position, at the moment when it attains the headpin, is used to compute a probability of striking. In practice, the parameters taken into account are the angle formed by the trajectory with the x-axis, so the angle of V_f as in Figure 1, and the ordinate of the ball y_f.

Both in [4] and [3], sample values of seasoned bowling players action on the ball are given and studied in simulations. They provide numerical values for initial velocity, angle and angular velocity, which are used as a starting point to determine initial conditions that can successfully lead to a strike. With V_0, ω_0 the norms of the initial velocities $\vec{V_0}$ and $\vec{\omega_0}$, this starting point is defined as :

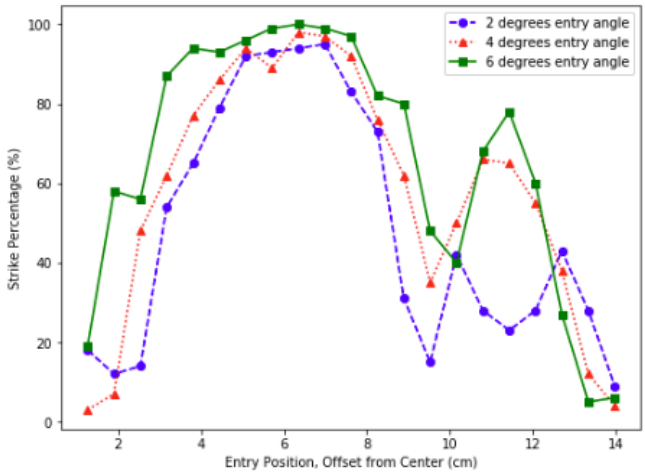


Fig. 3. Probability of making a strike as a function of the entry angle and position. Figure taken from [2] and [4].

$$\begin{cases} V_0 \simeq 8 \text{ m/s} \\ \omega_0 \simeq 30 \text{ rad/s} \\ \omega_{0z} < \omega_{Oy} \text{ and } \omega_{0z} < \omega_{Ox} \end{cases}$$

By studying the initial conditions and inertia matrix effect on the ball trajectory in the simulation, the values can be fine-tuned to suit a precise example. However, in order to leave some margin to the *Machine Learning controller*, the values are not optimized that way.

II. Simulation environment

A. Analytical model and main pipeline

In order to test the controllers, and how the inputs influence the ball trajectory, a precise simulation of a bowling lane is necessary. Thereby, the previous simplifying hypotheses on the center of mass needs to be cancelled, which allows to define the vector $\vec{r}_\Delta$ between the center of mass and the center of the ball. Here, the inertia tensor I is supposed diagonal, with its element I_{xx} the minimal moment of inertia, and I_{yy} and Izz unequal.

Taking these conditions into account, the work of Frohlich in [3] was largely exploited to determine the equations of motion of the ball. If the ball is sliding, the angular acceleration evolution is determined by :

$$(I_0 + I_{dev} + I^s_\Delta + I^s_{\Delta\Delta})\dot{\vec{\omega}} = \vec{\tau}_{dev} + \vec{\tau}_{fric} + \vec{\tau}^s_\Delta + \vec{\tau}^s_{\Delta\Delta} \quad (12)$$

And when the ball is rolling, the equation becomes :

$$(I_0 + I_{dev} + I^r_\Delta + I_{Roll})\dot{\vec{\omega}} = \vec{\tau}_{dev} + \vec{\tau}_{fric} + \vec{\tau}^r_\Delta + \vec{\tau}^r_{\Delta\Delta} \quad (13)$$

Where the torques and inertia matrices defined depend on the state of the ball, $\Delta\vec{r}$, R_b and μ in the sliding case. Their expression are detailed in the Annex and are taken from [3].

Using the Sympy Python library, the analytical expressions of the needed values was translated as functions of time. Then, using a 4$^{\text{th}}$ order Runge-Kutta as an integration method, the following pipeline is used for the simulation:

- The rotation matrix is integrated from $\vec{\omega}$, and used to compute I and $\Delta\vec{r}$.
- $\vec{s}$ defined in Eq. (4) and $\vec{\mu}$ defined in Eq. (5) are computed using the last values of $\dot{\vec{r}}$ and $\vec{\omega}$.
- Knowing if the ball is sliding or rolling, the value of $\vec{\omega}$ is integrated.
- Finally, using a straightforward integration when rolling and a double one when sliding, the position of the ball $\vec{r}$ is computed.

B. Some results and interpretation

The initial condition used as input of the simulation are :

$$\begin{cases} y_0 = 0 \text{ m} \\ \theta_0 = -1 \text{ °} \\ V_0 = 8 \text{ m/s} \end{cases} \quad \begin{cases} \omega_{0_x} = -30 \text{ rad/s} \\ \omega_{0_y} = -30 \text{ rad/s} \\ \omega_{0_z} = 10 \text{ rad/s} \end{cases}$$

Where θ_0 is the angle formed between $\vec{V_0}$ and the x-axis.

To create a realistic bowling ball, the data from [1] is used to inspire the parameters of the ball, which are:

$$\begin{cases} m = 6.80 \text{ kg} \\ I_{xx} = 0.0270 \text{ kg.m²} \\ I_{yy} = 0.0274 \text{ kg.m²} \\ I_{zz} = 0.0281 \text{ kg.m²} \end{cases}$$

Given these parameters, the resulting simulated trajectory is visualized in the figure 4 in true scale, and in figure 5 with a different scale.

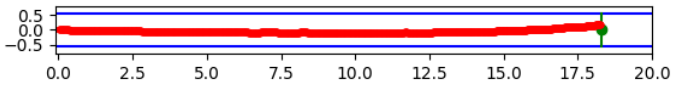


Fig. 4. Simulating a trajectory (in red), from the starting line to the headpin (in green) ordinate.

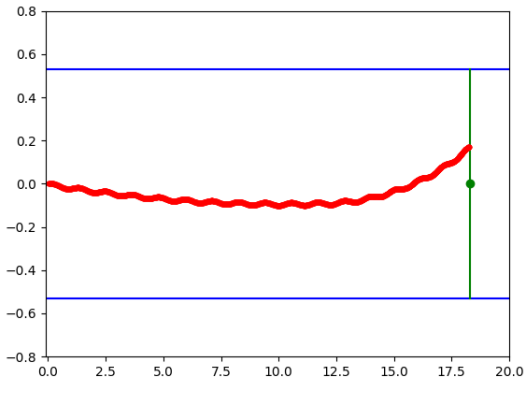


Fig. 5. Same results as in Fig. 4, with a different scale.

In Figure 4, the ball trajectory seems realistic. However, some oscillations orthogonal to the main trajectory are noticeable in Figure 5, with a period and amplitude of approximately 1.2 and 0.005 meters. The value traced in this graph is the evolution of the center of the ball position, which is not supposed to oscillate in this way. The leading potential cause for this phenomenon would be the computation of the center of the ball from the position of its center of mass. Indeed, this type of trajectory is likely for the center of mass due to the complex precession motion of the ball. Also, the computation of $\vec{r}_\Delta$ is bound to diverge from its true value, due to the successive integrations.

III. Learning Residuals

The second part of the final controller is a machine learning algorithm, used to better the trajectory of the bowling ball. It computes residuals to the initial conditions given by the previous part, the *physics-based* controller. Hence, the algorithm has to output 6 decimal numbers, each with different minimum and maximum boundaries.

To approach this problem, it was assimilated to a *k*-armed bandit problem. Meaning that the agent has a given situation before him, which is the set of initial conditions and its prior knowledge of the system, and has to choose one action between a set of possible actions. Once this action is taken, the system outputs a *reward*, allowing the agent to evaluate its prior choice.

This type of problem falls into Reinforcement Learning (RL). To define this framework and build a RL-algorithm, we rely on the work of Sutton and Barto in [6].

Due to successive time and programming issues, it has been impossible to properly train a RL model. This section exposes the choices that were made and offers some perspectives on how to proceed.

A. Rewards

The rewards given to the model are detailed in the table I. The event of a strike is determined using the values found in the Figure 3: given an entry angle and ordinate, a strike probability is computed from the reference values of the graph, and then the event is evaluated using the random package of the Numpy Python library. Every other situation listed can be seen as conditions on the position of the ball.

Situation of the ball	Reward
Gutter	-100
Pins attained but offset from the center ¿ 14cm	0
Pins attained and offset ¡= 14 cm	10
Pins attained, small offset and strike	100

TABLE I
REWARD CORRESPONDING TO DIFFERENT FINAL STATES OF THE BALL

B. Actions

The principal problem in the conception of this model, was that the classical Markov Decision Process terminology, that is used to define a RL-problem as in [6], accounts for a discrete set of actions, possible for the agent from a given state. When it would be ideal to learn the residuals as decimal values, it was decided to split the interval of possible values for each value into 8 possible values, see Table II.

Value	Min	Max	Delta
y_0 (cm)	-15	15	3.75
θ_0 (°)	-5	5	1.25
V_0 (m/s)	-2	2	0.5
ω_{x_0} (rad/s)	-20	20	5
ω_{y_0} (rad/s)	-20	20	5
ω_{z_0} (rad/s)	-6	6	1.5

TABLE II
BOUNDARIES OF THE DISCRETE SET OF POSSIBLE ACTIONS

Different strategies to choose between these actions were considered. In the case of a Q-Learning Model, aiming to maximize the reward over a succession of actions, it was decided to output 48 values, divide them in 6 subsets and choose the maximum for each one. In this way, an index for the corresponding action for each value can be obtained. Another solution could use one or multiple models to compute each values successively.

However, this method failed because of the reason mentioned earlier in this Section. The simulation time and some incompatibility between making multiple choices at the same time and the program used made the research fall behind schedule.

Conclusion

In this article, a superficial study of numerical values taken from bowling players was used to determine initial conditions for a throw. Using realistic motion equations determined in [3], a Python simulation is built and tested. The research done tried to build a dual controller, base both on a simplified analytical study of the motion equations and on Reinforcement Learning. However, some complication in the integration of said equations and complications on the RL models implementation and training were critical to this research.

Perspectives:

- Physical controller needs bettering, if possible, by finding an analytical solution and testing it in both a simplified and complete simulation, or a complete numerical study to create an abacus.
- A more powerful calculator would be useful to have more precise integration and test the simulation, also parallelization would quicken the computing, training and testing times.
- A further search for an adequate Machine Learning algorithm is needed.

Annex

Expression of $\dot{\vec{\omega}}$ in the sliding case

The matrix $I = I_0 + I_{dev}$ is the inertia matrix expressed in the world coordinate system. I_0 is its diagonal part, and I_{dev} the non-diagonal part. The torque $\vec{\tau}_{dev}$ stems from the development of (3), as $\vec{\tau}_{dev} = (I_{dev}\vec{\omega}) \times \vec{\omega}$.

We then have terms stemming from the expression of $\Delta\vec{r}$.

$$I^s_\Delta = R_b \begin{pmatrix} r_{\Delta_y}\mu_y & -r_{\Delta_x}\mu_y & 0 \\ r_{\Delta_y}\mu_x & r_{\Delta_x}\mu_x & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$I^s_{\Delta\Delta} = \begin{pmatrix} r_{\Delta_y}(r_{\Delta_y} - r_{\Delta_z}\mu_y) & -r_{\Delta_x}(r_{\Delta_y} - r_{\Delta_z}\mu_y) & 0 \\ r_{\Delta_y}(r_{\Delta_z}\mu_x - r_{\Delta_x}) & -r_{\Delta_x}(r_{\Delta_z}\mu_x - r_{\Delta_x}) & 0 \\ r_{\Delta_y}(r_{\Delta_x}\mu_y - r_{\Delta_y}\mu_x) & -r_{\Delta_x}(r_{\Delta_x}\mu_y - r_{\Delta_y}\mu_x) & 0 \end{pmatrix}$$

Finally, various torques are defined :
$\vec{\tau}_{fric} = -gR_b\vec{z} \times \vec{\mu}$,
$\vec{\tau}^s_\Delta = g\vec{r}_\Delta \times \mu - a_w R_b \vec{z} \times \mu$, and
$\vec{\tau}^s_{\Delta\Delta} = a_w \vec{r}_\Delta \times \mu$.
where
$a_\omega = [(\vec{\omega} \times \vec{r}_\Delta) \times \vec{\omega}]_z$

Expression of $\dot{\vec{\omega}}$ in the rolling case

Keeping the same definitions for I_0, I_{dev}, $\vec{\tau}_{dev}$ and $\vec{\tau}_{fric}$ as earlier, the remaining matrices and torques are :

$$I_{Roll} = \begin{pmatrix} R_b^2 & 0 & 0 & \\ 0 & R_b^2 & 0 & 0 \\ 0 & 0 & 0 & \end{pmatrix}$$

$$I^r_\Delta = \begin{pmatrix} r^2_{\Delta_y} + r^2_{\Delta_z} - 2r_{\Delta_z}R_b & -r_{\Delta_x}r_{\Delta_y} & -r_{\Delta_x}r_{\Delta_z} + r_{\Delta_x}R_b \\ -r_{\Delta_x}r_{\Delta_y} & r^2_{\Delta_x} + r^2_{\Delta_y} - 2r_{\Delta_z}R_b & -r_{\Delta_y}r_{\Delta_z} + r_{\Delta_y}R_b \\ -r_{\Delta_x}r_{\Delta_z} + r_{\Delta_x}R_b & -r_{\Delta_y}r_{\Delta_z} + r_{\Delta_y}R_b & r^2_{\Delta_x} + r^2_{\Delta_y} \end{pmatrix}$$

$\vec{\tau}^r_\Delta = g\vec{r}_\Delta \times \vec{z} + R_b[||\vec{\omega}||^2 \vec{r}_\Delta \times \vec{z} + (\vec{\omega} * \vec{r}_\Delta) * \vec{z} \times \vec{\omega}]$, and
$\vec{\tau}^r_{\Delta\Delta} = (\vec{\omega} * \vec{r}_\Delta) * \vec{\omega} \times \vec{r}_\Delta$.

REFERENCES

[1] ABSOLUTE POWER. URL https://www.stormbowling.com/absolute-power-bbmvaw12.

[2] Donald Benner, Nicole Mours, and Paul Ridenour. Pin Carry Study: Bowl Expo 2009. 2009.

[3] Cliff Frohlich. What makes bowling balls hook? *American Journal of Physics*, 72(9):1170–1177, September 2004. ISSN 0002-9505. doi: 10.1119/1.1767099. URL https://doi.org/10.1119/1.1767099.

[4] Simon Ji, Shouzhuo Yang, Wilber Dominguez, and Cacey Bester. Using Physics Simulations to Find Targeting Strategies in Competitive Bowling, October 2022. URL http://arxiv.org/abs/2210.06753. arXiv:2210.06753 [physics].

[5] Franco Normani. Physics Of Bowling. URL https://www.real-world-physics-problems.com/physics-of-bowling.html.

[6] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: an introduction*. Adaptive computation and machine learning. The MIT Press, Cambridge (Mass.), 2nd edition edition, 2018. ISBN 978-0-262-03924-6.

[7] Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. TossingBot: Learning to Throw Arbitrary Objects with Residual Physics, May 2020. URL http://arxiv.org/abs/1903.11239. arXiv:1903.11239 [cs, stat].

Adam GOUX--GATEAU

UE 5.1 Research initiation

DESCRIPTION OF CHAOS WITH THE LYAPUNOV EXPONENT IN NONLINEAR EQUATIONS

March 1st 2024

Contents

1 Introduction

Lately, I read an article presenting a description of chaos in the famous system of the double pendulum. It is quoted as "[**5**]" in the following report. Finding quite fascinating the notion of chaos, I decided to explore it myself, and to look for other chaotic systems, and ways to mesure the chaos.

In mathematical terms, chaos refers to a state exhibited by certain dynamical systems characterized by sensitive dependence on initial conditions, unpredictability, and the absence of long-term regularity. A dynamical system is considered chaotic if it possesses sensitivity to initial conditions, wherein small changes in the initial conditions lead to significantly divergent trajectories in phase space. This sensitivity is mathematically represented by positive Lyapunov exponents [**1**], indicating exponential separation of nearby trajectories. Additionally, chaotic systems exhibit topological mixing, where trajectories densely fill phase space, lacking periodic orbits or stable fixed points. Nonlinear dynamics play a crucial role in the manifestation of chaos, as the interactions between variables in nonlinear equations introduce complexities that give rise to chaotic phenomena.

Nonlinear equations are instrumental in generating chaotic behavior within dynamical systems. Unlike linear equations, which often lead to predictable and stable dynamics, nonlinear equations introduce complexities that foster chaotic phenomena. In nonlinear systems, interactions between variables are not simply additive or proportional, but instead involve feedback loops, bifurcations, and other nonlinear effects. These nonlinearities amplify sensitivity to initial conditions, resulting in chaotic behavior characterized by erratic and seemingly random trajectories in phase space. Examples of chaotic systems governed by nonlinear equations include the Lorenz system, the logistic map, and the double pendulum. Understanding and analyzing chaos in nonlinear equations are essential for elucidating the behavior of complex systems across various scientific disciplines, including physics, biology, and engineering.

In this article, I will briefly illustrate chaos theory with a simple example, and then see the application in real systems : the double pendulum and Chua circuit

All the code mentioned is available here :
https://github.com/Gougaaate/chaospy

2 Mathematical description of chaos

Chaos theory is above all a branch of mathematics. I will develop in this part the mathematical aspects of chaos, especially the Lyapunov exponent.

The Lyapunov exponent of a dynamical system, denoted by λ, is a real number that characterizes the system's behavior. It measures the average rate of divergence or convergence of nearby trajectories in phase space. This divergence or convergence indicates how sensitive the system is to initial conditions**[2]**. A positive Lyapunov exponent indicates exponential separation of trajectories, implying chaotic behavior, while a negative exponent suggests convergence towards a stable equilibrium. If the initial separation in phase space is δ_0, we have the relation :

$$|\delta(t)| = |\delta_0|e^{\lambda t} \tag{1}$$

λ is defined as follows :

$$\lambda \triangleq \lim_{t\to\infty} \lim_{|\delta_0|\to 0} \frac{1}{t} \ln \frac{|\delta(t)|}{|\delta_0|} \tag{2}$$

To evaluate the chaotic aspect of dynamical systems, I decided to code my own calculator of Lyapunov exponent **[3]**.

I took the example of the famous logistic equation **[4]**:

$$x_{n+1} = \mu x_n(1 - x_n);\ x_0 \in [0;\ 1] \tag{3}$$

The equation describes the progression of a population denoted as x, starting with an initial quantity x_0. The parameter μ ranges between 0 and 4, exerting significant influence on the population's dynamics. When $\mu \leq 1$, the population dwindles to extinction. For $\mu \in [2, 3]$, the population stabilizes around a specific value, albeit exhibiting some fluctuations. However, with $\mu > 3$, intriguing phenomena emerge: the sequence might oscillate between distinct values, following a pattern of doubling (2, 4, 8, etc.), or it could plunge into complete chaos. In the `logistic.py` file, a map illustrates the sequence's behavior, indicating the intervals between oscillating values and reflecting the complexity of the population's dynamics :

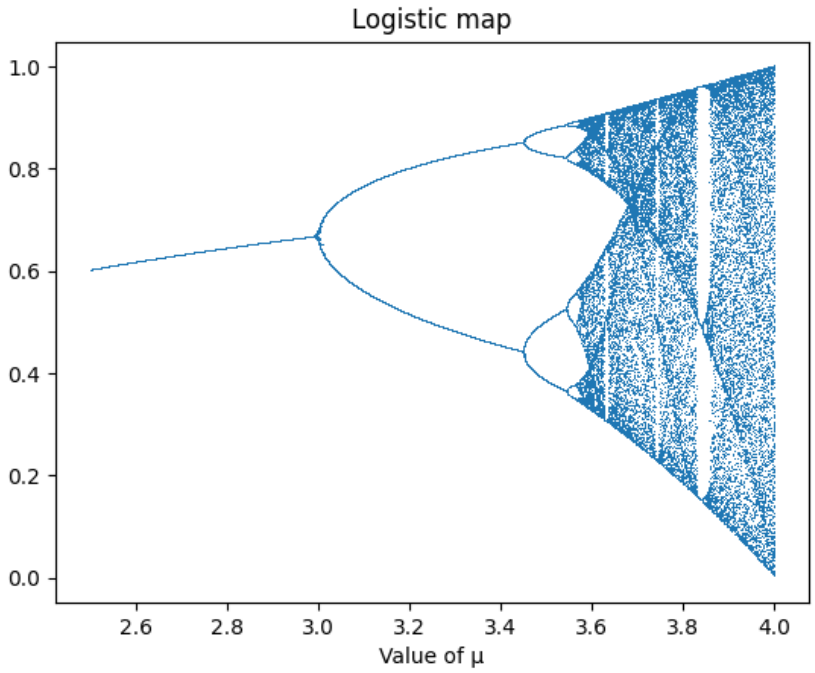


Figure 1: Bifurcation diagram

As μ increases from 0 to 4, the bifurcation diagram reveals the different types of behavior exhibited by the logistic map. At lower values of μ, the population typically converges to a stable equilibrium or undergoes periodic oscillations. As μ increases, the diagram shows bifurcation points where the behavior of the system changes. Each bifurcation means that the sequence oscillate between several values. We clearly see chaos emerging if $\mu > \approx 3.56$, with no distinguishable pattern.

I computed the Lyapunov exponent for several values of the sequence, to confirm the chaotic behaviour :

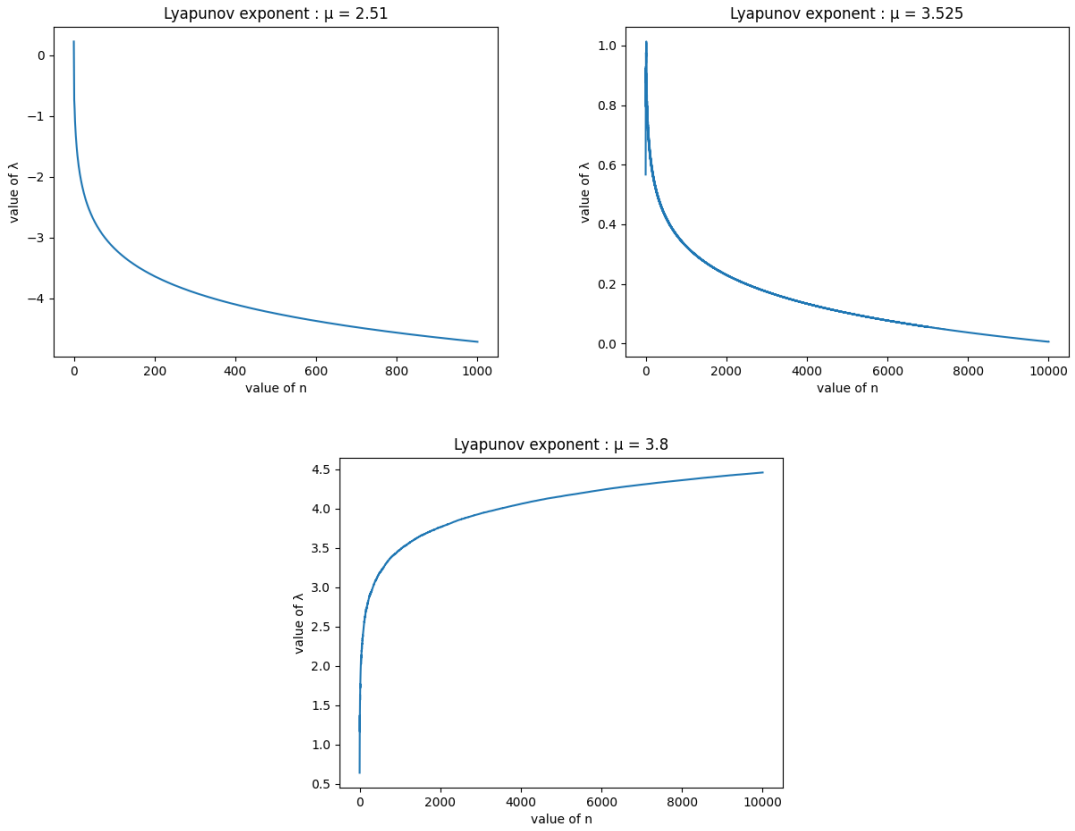


Figure 2: Evolution of λ for $\mu = 2.51$, 3.525 and 3.8

To compute λ, I used a formula for discrete systems. If we have $x_{n+1} = f(x_n)$, then $\lambda = \lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \ln |f'(x_i)|$. I computed the successive values of λ to see the tendance : for $\mu < 3$, $\lambda < 0$ so the system is stable, for $\mu \approx 3.53$, $\lambda \approx 0$, so the system is slightly unstable, for $\mu > 3.6$, $\lambda > 0$: chaos begins. This is exactly the result expected for this sequence.

3 Application in physics

As seen before, chaos can appear anywhere as far as nonlinear equations are involved. To illustrate this, I performed several simulations of well known or not real systems with a chaotic behaviour.

3.1 Double pendulum

First of all, the double pendulum might be the example that enables the best visualisation of chaos. Here are the equations :

$$\begin{cases} m_2\left(g\sin(\theta_1) + l_2\left((\theta_2')^2\sin(\theta_1-\theta_2) + \theta_2''\cos(\theta_1-\theta_2)\right) + l_1\theta_1''\right) + m_1\left(g\sin(\theta_1) + l_1\theta_1''\right) = 0 \\ g\sin(\theta_2) + l_1\left(\theta_1''\cos(\theta_1-\theta_2) - (\theta_1')^2\sin(\theta_1-\theta_2)\right) + l_2\theta_2'' = 0 \end{cases} \tag{4}$$

They are clearly non linear, but that does not imply a chaotic behaviour. The Lyapunov exponent is once again a great mean of evaluating the chaos. There are actually 4, because this is a 4-state dynamic system ($\theta_1, \omega_1, \theta_2, \omega_2$ where θ and ω represent the angle and the angular speed of each mass). I encountered a problem of computational cost : for complex equations like this and multi-dimensional problems, the computation of Lyapunov exponent is really intensive to be accurate, and I could not implement a reliable method. Hopefully, this had already been done [**5**] :

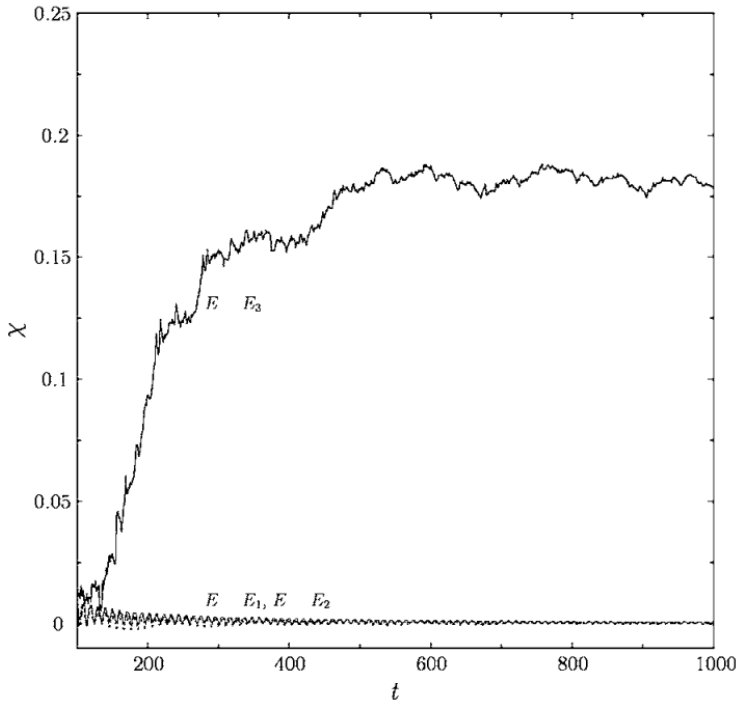


Figure 3: Lyapunov exponent of a double pendulum (T. Stachowiak, T. Okada)

As you can see, the value is positive, which explains the chaotic behaviour of such a system. To visualize it better, I performed a simulation of two double pendulums with very close initial states in `double_pendulum.py`. After a few seconds, here is the state of the system :

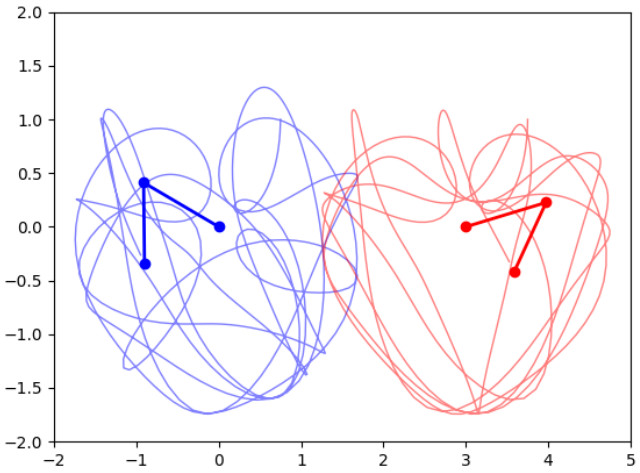


Figure 4: Divergence of two double pendulums

The trajectories have already diverged after 4 seconds, the difference was 0.05° for θ_2 in the initial state.

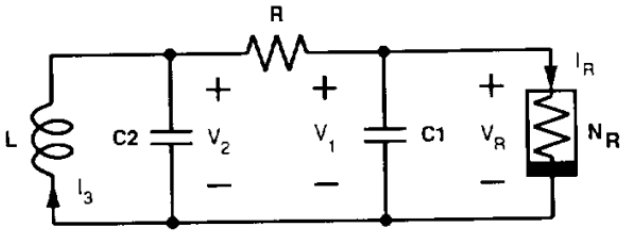
3.2 Electric circuits

Electric circuits, governed by deterministic laws of physics, can exhibit chaotic behavior under specific conditions. Typically observed in circuits featuring feedback mechanisms, chaotic phenomena emerge when the circuit's parameters are finely tuned or subjected to external disturbances [**6**]. In such scenarios, minor fluctuations in input signals or circuit parameters yield disproportionate alterations in the system's output. This sensitivity to initial conditions results in the erratic and quite unpredictable nature of chaos. I will only consider simple circuits made from standard components, such as resistors, inductance, diode...

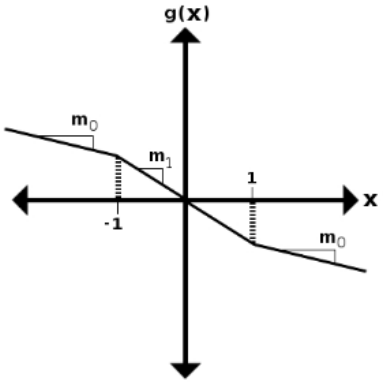
Before displaying chaotic behaviour, a circuit must satisfy three conditions :

- At least one non linear element (this is quite obvious, the Lyapunov exponent for linear equations will almost always be < 0)
- At least one active resistor
- At least 3 components storing energy, such as capacitors or inductors.

The simplest circuit verifying thess conditions is called Chua circuit [**7**]. It is composed of a resistor, an inductor, two capacitors and one "Chua's diode", which follows a piecewise-linear equation.



(a) Chua circuit



(b) Chua's diode equation

The equation of Chua's diode is presented here :

$$g(x) = \begin{cases} m_0 x + m_0 - m_1 & \text{if } x \leq -1 \\ m_1 x & \text{if } -1 \leq x \leq 1 \\ m_0 x + m_1 - m_0 & \text{if } 1 \leq x \end{cases} \tag{5}$$

A simplified version of the equations is easily findable, but the real equation using the values of the components is almost never written. That is why i applied Kirchoff's laws to find the nonlinear equation ruling the circuit. I obtained :

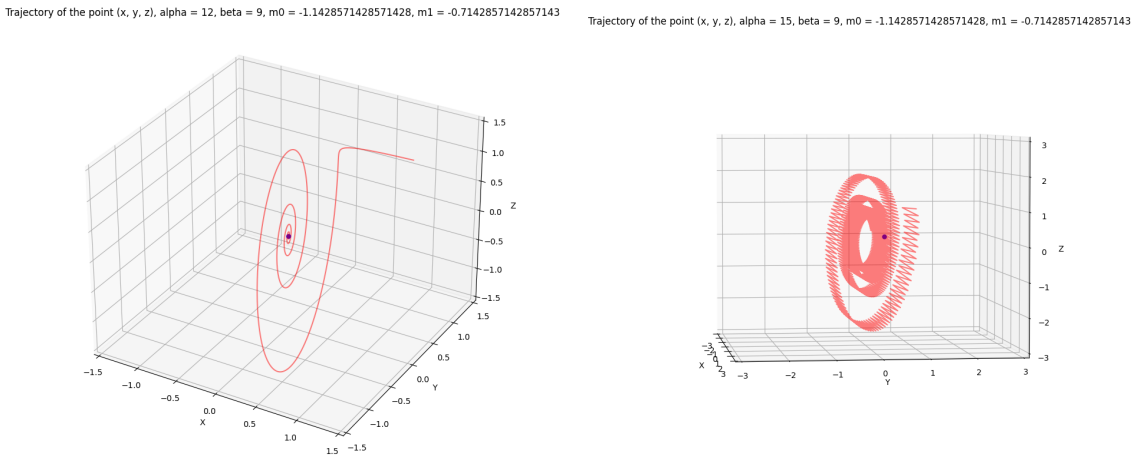
$$\begin{cases} \dot{V_1} &= \left(\dfrac{1}{RC_1}\right)[(V_2 - V_1) - R.g(V_1)] \\ \dot{V_2} &= \left(\dfrac{1}{RC_2}\right)(V_1 - V_2 + R.i_L) \\ i_L &= \dfrac{-V_2}{L} \end{cases} \tag{6}$$

With simplifications and rescaling of V_1, V_2, and i_L, we can finally obtain dimensionless Chua's equations:

$$\begin{cases} \dot{x} &= \alpha[y - \Phi(x)] \\ \dot{y} &= x - y + z \\ \dot{z} &= -\beta y \end{cases} \tag{7}$$

With : $\Phi(x) \triangleq x + g(x) = m_1 x + \dfrac{1}{2}(m_0 - m_1)(|x+1| - |x-1|)$

In the file `chua.py`, I plotted the evolution of (x, y, z) to find some attractors, here are the most interesting I could display :



(a) (b)

Figure 6: 2 strange attractors

Now, the interesting part is to compute the Lyapunov exponent of this circuit. I encountered the same problem of computational cost as before, but I came up with the idea of comparing two close trajectories. I simply evaluated the distance between the two during the simulation.

My initial states were $x_0 = y_0 = z_0 = 1$, and $x_1 = y_1 = z_1 = 1.01$

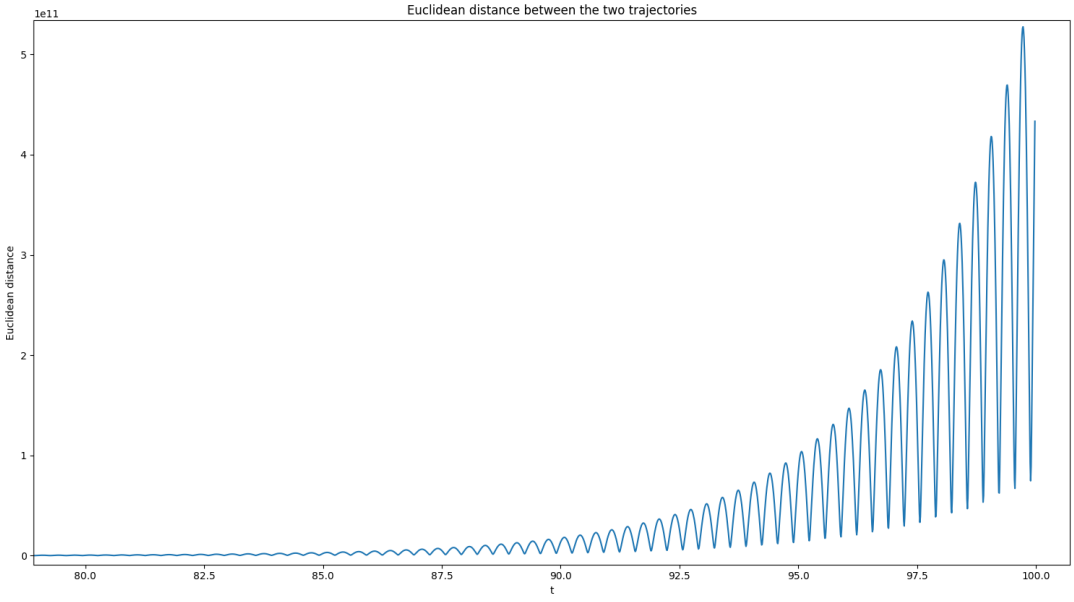


Figure 7: Divergence of the distance between the two close trajectories

As you can see, the distance slightly oscillates around 0.01 and then diverge. This does not depend from the integration method : Euler with step from 0.05 to 0.0001, and RK4 gave almost the same results. This is an other way of evaluating the chaos of a system with only a few dimensions.

4 Conclusion

While exploring techniques to assess chaos, I encountered several efficient methods, with the primary one being the Lyapunov exponent. As the complexity of the problem grew beyond the capabilities of my own laptop, I began considering alternative approaches to demonstrating the presence of chaos. One such approach involves leveraging the Euclidean distance.

This concept finds relevance in the field of robotics for several reasons. Firstly, chaos can emerge in feedback-controlled loops [**8**] when nonlinear ordinary differential equations (ODEs) are present. In the context of robot guidance, it's imperative to mitigate chaos to prevent unintended behavior. However, chaos might also prove beneficial. For instance, in path planning without mapping [**8**], deterministic chaos can be harnessed using chaotic dynamical systems. By utilizing chaotic attractors, algorithms for obstacle avoidance and pathfinding can be developed.

This area of research remains highly promising and holds numerous applications across various scientific disciplines, including robotics.

5 Bibliography

[1] : *Grandes déviations d'exposants de Lyapunov dans les systèmes étendus*, T. Laffargue, Mécanique statistique [cond-mat.stat-mech]. Université Paris Diderot (Paris 7), 2015. Français. ffNNT : ff.

[2] : M. Amiri, M. Dehghani, A. Khayatian and M. Mohammadi, *Lyapunov Exponent based Stability Assessment of Power Systems*, 2019 6th International Conference on Control, Instrumentation and Automation (ICCIA), Sanandaj, Iran, 2019, pp. 1-5, doi: 10.1109/ICCIA49288.2019.9030854.

[3] : *Numerical Calculation of Lyapunov Exponents*, M. Sandri, University of Verona, Italy.

[4] : Verhulst and the logistic equation (1838). In: *A Short History of Mathematical Population Dynamics*, N. Bacaër (2011). Springer, London. https://doi.org/10.1007/978-0-85729-115-8_6.

[5] : *A numerical analysis of chaos in the double pendulum, p.7* Tomasz Stachowiak, Toshio Okada.. Chaos, Solitons and Fractals, 2006, 29 (2), pp.417 - 422. ff10.1016/j.chaos.2005.08.032ff. ffhal-01389907f.

[6] : *Chaos and its applications*, Kazuyuki Aihara, Institute of Industrial Science, University of Tokyo, 4-6-1 Komaba, Meguro-ku, Tokyo 153-8505, Japan.

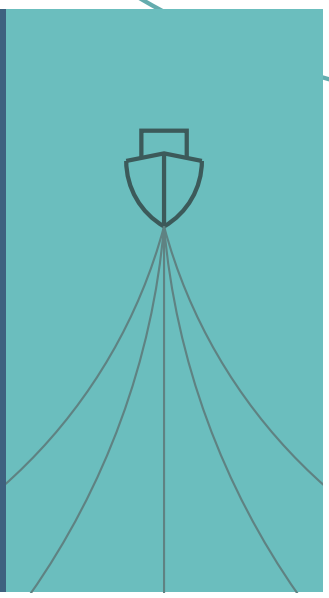
[7] : *Chaos in Electronic Circuits*, T. Matsumoto, IEEE, 1987.

[8] : *Applications of Chaotic Dynamics in Robotics, pp.3,7,8*, Xizhe Zang, Sajid Iqbal, Yanhe Zhu, Xinyu Liu, and Jie Zhao, International Journal of Advanced Robotic Systems, 2017.

29 février 2024

Localisation Intérieure par Analyse des Ondes WiFi

Introduction à la Recherche 2024



GROS Louis-Nam

ENSTA BRETAGNE

Abstract

Cet article propose une nouvelle méthode pour la localisation précise de robots en environnement intérieur, exploitant les signaux WiFi omniprésents. En analysant les ondes émises par les points d'accès WiFi, nous développons une approche permettant de trianguler avec précision la position du drone. Cette méthode, basée sur des algorithmes avancés de traitement du signal, offre une alternative efficace aux systèmes de localisation traditionnels, en particulier dans les contextes où l'usage de GNSS est limité ou inexistant.

Keywords

— Localisation intérieure
— WiFi
— Atténuation des signaux
— Couverture du signal
— Multilatération
— Algorithme BPSA (Binary Particle Swarm Optimization)
— Optimisation de Placement d'antennes
— Dilution Of Precision (DOP)
— Erreur quadratique moyenne (RMSE)
— Simulation 3D
— Infrastructures technologiques modernes

Table des matières

Introduction

La localisation en intérieur est un domaine de recherche dynamique et essentiel qui sert de fondement à de nombreuses applications modernes, telles que la navigation intérieure, la robotique mobile, et le suivi d'objets. Avec la prévalence accrue des appareils mobiles et la nécessité d'une connectivité omniprésente, les technologies de localisation intérieure sont devenues un élément clé de l'infrastructure technologique contemporaine. Cet article passe en revue les technologies et les méthodologies actuelles en matière de localisation intérieure, mettant en lumière leurs forces, leurs limites, et les contextes dans lesquels elles sont le mieux adaptées.

Les technologies telles que l'infrarouge, les ultrasons, le RFID, Bluetooth, ZigBee, WiFi et l'Ultra Wide Band sont explorées en profondeur [3], révélant un paysage diversifié de solutions adaptées à une gamme variée de cas d'utilisation. Chaque technologie présente un compromis entre la portée, la précision, le coût et la complexité d'intégration, soulignant l'importance de choisir la bonne solution pour les besoins spécifiques de chaque application. Par ailleurs, l'étude présente un éventail de méthodes et d'algorithmes de localisation, de la simple localisation par zone à des approches plus sophistiquées telles que la triangulation, la multilatération, le fingerprinting, et la navigation à l'estime.

1 État de l'art

1.1 Technologies de localisation en intérieur

1.1.1 Infrarouges

Les infrarouges (IR), invisibles à l'œil humain de par leur longueur d'onde supérieure à celle de la lumière visible, sont largement utilisés en domotique et robotique, notamment pour la commande à distance et la détection d'obstacles. Ils se divisent en approches active et passive. L'approche active utilise des diodes électroluminescentes (LED) émettant des signaux IR, captés par des photo-diodes ou caméras IR pour la localisation[2]. La méthode passive repose sur la thermographie, exploitant la radiation naturelle des objets. Malgré leur faible coût et simplicité, les signaux IR nécessitent une visibilité directe entre émetteur et récepteur et sont limités à une portée de 5-10 mètres, avec une sensibilité aux sources lumineuses ambiantes.

1.1.2 Ultrasons

Les ultrasons, vibrations acoustiques au-delà de l'audible, sont générés par des transducteurs exploitant l'effet piézoélectrique. Utilisés principalement entre 20-50 KHz, ils offrent l'avantage d'une propagation lente, facilitant la mesure temporelle pour la localisation et permettant une précision sub-millimétrique. Cependant, leur portée est limitée à 10-15 mètres et leur performance peut être altérée par les conditions environnementales telles que la température, l'humidité et les courants d'air.

1.1.3 RFID

La RFID, technologie sans fil basée sur la lecture de transpondeurs ou tags, varie en portée selon la fréquence utilisée, avec des différences en termes de pénétration matérielle et de débit de données. Les systèmes à courte portée utilisent un couplage inductif, où les tags passifs tirent leur énergie par induction. Les tags actifs possèdent leur propre source d'alimentation. Malgré la disponibilité de lecteurs bon marché, leur intégration réseau peut engendrer des coûts additionnels.

1.1.4 Bluetooth

Bluetooth, technologie sans fil utilisant l'étalement de spectre à sauts de fréquence, opère dans la bande ISM autour de 2.4 GHz. Elle permet la coexistence avec d'autres technologies comme le WiFi ou le ZigBee grâce à des sauts de fréquence, avec des équipements classés selon leur portée et puissance d'émission.

1.1.5 ZigBee

ZigBee, basé sur le standard IEEE 802.15.4, est simple d'utilisation grâce à sa pile protocolaire légère. Avec une autonomie de plusieurs années sur piles classiques, il convient aux réseaux de capteurs et aux applications de contrôle, malgré un faible débit et des interférences potentielles avec d'autres technologies sans fil.

1.1.6 WiFi

Le WiFi, ou IEEE 802.11, utilise les bandes ISM de 2.4 GHz ou 5 GHz. Les systèmes de localisation basés sur WiFi exploitent principalement les trames de balisage émises par les points d'accès pour la localisation, avec des informations comme le SSID pour l'identification du réseau [4].

1.1.7 Ultra Wide Band (UWB)

L'Ultra Wide Band (UWB) transmet des impulsions de très courte durée, avec une émission limitée sur le spectre de 6 à 9 GHz. Cette technologie permet une détection efficace des multi-trajets, bénéficiant de sa large bande pour une localisation précise. Elle supporte des approches actives et passives pour la localisation, bien que l'emploi d'un récepteur avec réseau d'antennes puisse complexifier son utilisation dans le grand public.

1.2 Méthodes et algorithmes de localisation

1.2.1 Localisation par zone

La localisation par zone estime la position en déterminant la zone où se trouve le mobile, basée sur la détection par des points de référence. Simple et peu coûteuse, mais avec une précision limitée aux zones de couverture définies.

1.2.2 Triangulation

La triangulation détermine la position par le recoupement des angles d'arrivée (AOA) des signaux émis par les points de référence. Nécessite au moins deux mesures AOA pour une localisation 2D.

1.2.3 Barycentre

Le barycentre est calculé comme suit:

$$\hat{r} = \sum_{i \in I} w_i r_i$$

où r_i sont les positions des points de référence, w_i les poids attribués à chaque point de référence, avec $\sum w_i = 1$. Cette méthode est simple et requiert peu de ressources.

1.2.4 Multilatération

La multilatération utilise les distances estimées entre le mobile et plusieurs points de référence pour déterminer la position. Pour une localisation 2D ou 3D, elle peut être exprimée par:

$$d_i = ||r - r_i||_2$$

où d_i est la distance estimée entre le mobile et le point de référence i, et r, r_i sont les positions du mobile et du point de référence, respectivement.

1.2.5 Fingerprinting

Le fingerprinting compare les mesures signal avec une base de données de "signatures" de signal pré-établie pour estimer la position. Cette méthode nécessite une phase d'apprentissage pour construire la base de données.

1.2.6 Navigation à l'estime

La navigation à l'estime estime le déplacement à partir d'un point connu, en utilisant un modèle récursif:

$$p_k = \begin{bmatrix} x[k] \\ y[k] \end{bmatrix} = \begin{bmatrix} x[k-1] \\ y[k-1] \end{bmatrix} + d[k] \begin{bmatrix} \sin(\psi_u[k]) \\ \cos(\psi_u[k]) \end{bmatrix}$$

où $d[k]$ est la distance parcourue et $\psi_u[k]$ l'orientation de déplacement à l'itération k.

1.2.7 SLAM (Simultaneous Localization and Mapping)

L'algorithme de localisation SLAM permet à un appareil de comprendre sa position dans un espace tout en construisant simultanément une carte de cet environnement inexploré. Il débute par l'estimation de la position initiale de l'appareil à l'aide de capteurs, puis procède à l'identification des points de repère environnants qui servent de références pour la navigation et la cartographie. À mesure que l'appareil se déplace, l'algorithme ajuste continuellement les estimations de sa position et de l'orientation en comparant les données sensorielles entrantes avec les points de repère déjà cartographiés, permettant ainsi de minimiser l'erreur de localisation. En parallèle, il met à jour et affine la carte de l'environnement en intégrant de nouvelles observations, ce qui contribue à une meilleure précision de localisation dans le temps. Le SLAM s'appuie sur des techniques complexes comme le filtrage de Kalman ou les graphes de pose pour optimiser à la fois la carte de l'environnement et la localisation de l'appareil, rendant possible la navigation autonome dans des espaces inconnus.

2 Localisation par technologie WiFi

2.1 Présentation de la technologie WiFi[1]

La technologie WiFi s'appuie sur la norme IEEE 802.11 pour fournir une connectivité sans fil haute performance à travers différentes bandes de fréquence, principalement 2.4 GHz et 5 GHz. Les protocoles de modulation comme OFDM et MIMO sont utilisés pour améliorer la fiabilité et le débit des communications sans fil. Dans les environnements intérieurs, où les signaux GPS sont souvent indisponibles ou imprécis, le WiFi devient un choix privilégié pour la localisation grâce à sa pénétration étendue et à la présence de nombreux points d'accès.

2.2 Stratégie retenue pour la localisation WiFi

Notre approche pour établir la localisation intérieure à l'aide de la technologie WiFi inclut plusieurs étapes clés:

1. Modélisation de l'atténuation de la puissance du signal des ondes WIFI.[4]
2. Identification du placement optimal avec l'algorithme BPSA des points d'accès WiFi (balises) pour maximiser la couverture tout en minimisant les interférences et les zones mortes.[3]
3. Utilisation de la trilatération pour calculer la position du robot. La multilatération est une méthode basée sur la mesure de la force du signal reçu (RSS) de trois balises ou plus pour déterminer la position précise d'un appareil.[2]
4. Réalisation de simulations en plaçant le robot dans 1000 positions générées aléatoirement dans l'espace couvert, afin d'évaluer la robustesse et la fiabilité de la méthode de localisation proposée.
5. Analyse comparative des résultats de localisation obtenus avec et sans la présence d'obstacles physiques, tels que les murs ou le mobilier, qui peuvent atténuer ou réfracter les signaux WiFi et ainsi impacter la précision de la localisation.

Ces étapes permettront d'évaluer l'efficacité de la localisation WiFi dans divers scénarios et de déterminer les meilleures pratiques pour l'implémentation dans des environnements réels.

2.3 Choix des paramètres de simulations

2.3.1 Paramètres matériels

Pour nos simulations, nous avons opté pour un bâtiment de dimensions 40m x 40m x 10m sans étages. Les balises WiFi utilisées sont des UniFi AC Mesh DS, choisies pour leurs spécifications adaptées à la localisation en intérieur. Ces balises sont connues pour leur portée étendue (140m maximum) et leur capacité à supporter des communications à haut débit, essentielles pour la précision de la localisation.

2.3.2 Génération d'obstacles[1]

Nous simulerons trois obstacles verticaux placés de manière aléatoire dans l'environnement pour évaluer l'impact des interférences physiques sur la précision de la localisation. Ces obstacles représentent des murs internes susceptibles d'atténuer ou de réfléchir les signaux WiFi, affectant ainsi le processus de trilatération. Chaque mur est modélisé comme ayant un indice de réfraction spécifique qui influencera la propagation des ondes WiFi selon le modèle d'atténuation choisi.

3 Modèle d'atténuation des ondes (log-distance)[4]

3.1 Mesure RSS (Received Signal Strength)

La mesure RSS est un indicateur clé de la puissance du signal WiFi perçu par les récepteurs. Le modèle log-distance, exprimé par la formule ci-dessus, permet d'estimer cette puissance en se basant sur la distance par rapport à l'émetteur et sur un coefficient d'atténuation qui varie selon l'environnement.

3.2 Modèle d'atténuation des ondes WiFi dans l'air d'une antenne WIFI

La diffusion des ondes WiFi dans l'air suit un modèle radial où la puissance du signal diminue avec la distance par rapport à la balise. Cela est représenté visuellement par un gradient de couleur allant du jaune vif au bleu foncé, indiquant une diminution de la force du signal de 100% au centre à des valeurs plus basses en périphérie. Cette distribution peut être modélisée mathématiquement par la formule du modèle log-distance. loi de puissance logarithmique, exprimée par la formule suivante où $P(d)$ est l'intensité du signal reçue à une distance d, $P(d_0)$ est l'intensité de référence du signal à une distance d_0, et n est le coefficient d'atténuation.

$$P(d) = P(d_0) - 10n \log_{10}\left(\frac{d}{d_0}\right) \tag{1}$$

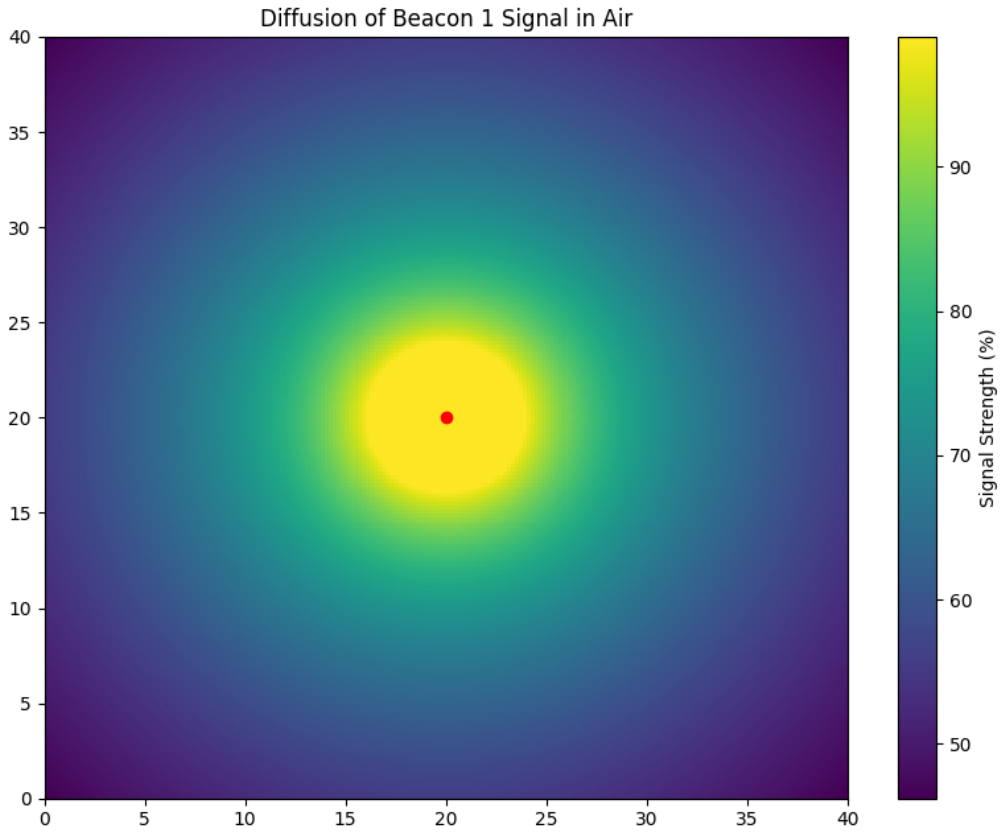


FIGURE 1 – Modèle d'atténuation des ondes WiFi dans l'air d'une antenne WIFI

3.3 Modélisation de l'atténuation des ondes WiFi à travers les obstacles[4]

Les obstacles tels que les murs en béton introduisent une atténuation supplémentaire du signal WiFi. Cette atténuation est reflétée par une augmentation du coefficient n dans le modèle log-distance. Les simulations montrent une zone d'ombre derrière l'obstacle où la force du signal est considérablement réduite, ce qui correspond à un changement de couleur vers le vert ou le bleu dans l'image. Cette

atténuation peut être intégrée dans la modélisation pour prédire la portée et la qualité du signal dans des conditions réelles.

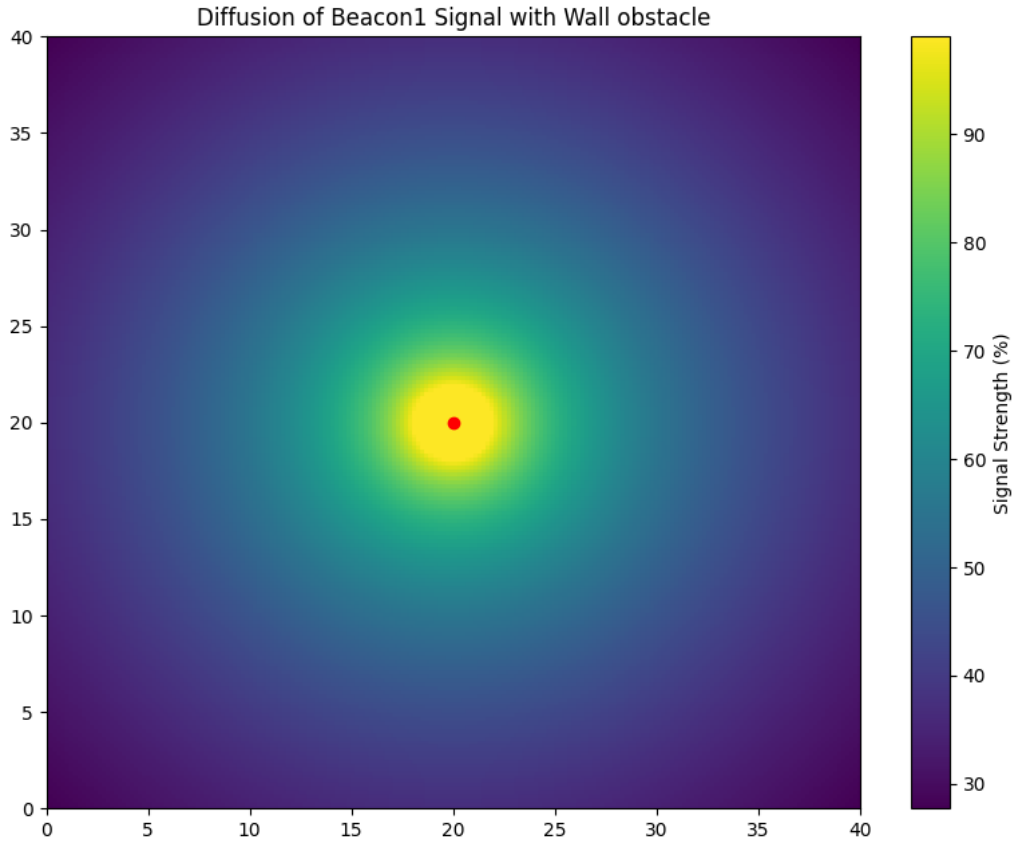


FIGURE 2 – Modèle d'atténuation des ondes WiFi dans l'air d'une antenne WIFI

4 Localisation par multilatération

4.1 Théorie de la multilatération[2]

La multilatération détermine la position d'un point en mesurant sa distance à d'autres points de référence connus. En deux dimensions, la position du point est l'intersection des cercles centrés sur ces points de référence. Pour la trilatération, si nous avons trois points de référence (x_1, y_1), (x_2, y_2), et (x_3, y_3) avec les rayons r_1, r_2, et r_3 respectivement, les équations des cercles sont données par:

$$(x - x_1)^2 + (y - y_1)^2 = r_1^2 \tag{2}$$

$$(x - x_2)^2 + (y - y_2)^2 = r_2^2 \tag{3}$$

$$(x - x_3)^2 + (y - y_3)^2 = r_3^2 \tag{4}$$

La position (x, y) du point cible est le point qui satisfait ces trois équations simultanément.

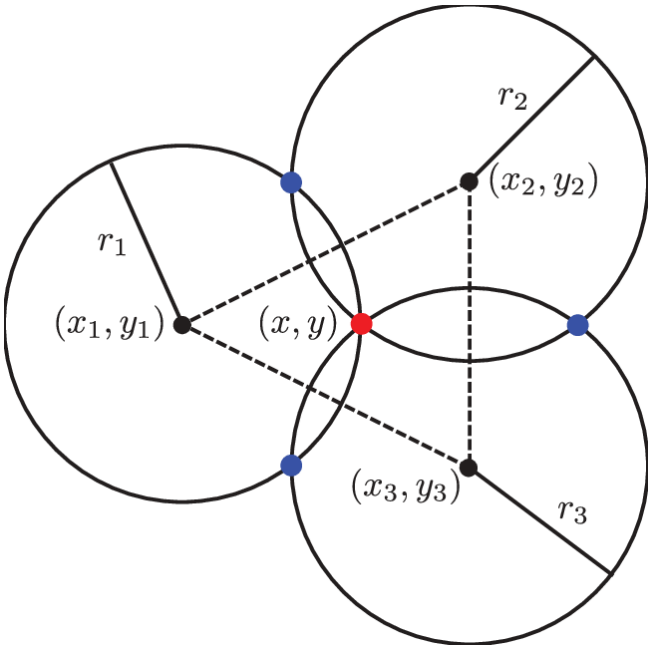


FIGURE 3 – Trilatération

4.2 Implémentation avec l'atténuation des ondes WiFi

Dans la simulation, un "bruit" est introduit pour modéliser l'incertitude de la mesure des distances due à l'atténuation du signal WiFi. Ce bruit est proportionnel au carré de la perte de puissance du signal, simulant ainsi une précision de mesure qui diminue avec la distance par rapport à la balise. La trilatération est ensuite appliquée en prenant en compte cette incertitude variable, ce qui peut conduire à une estimation de position qui n'est pas à l'intersection exacte des trois cercles mais dans une région approximative autour de celle-ci.

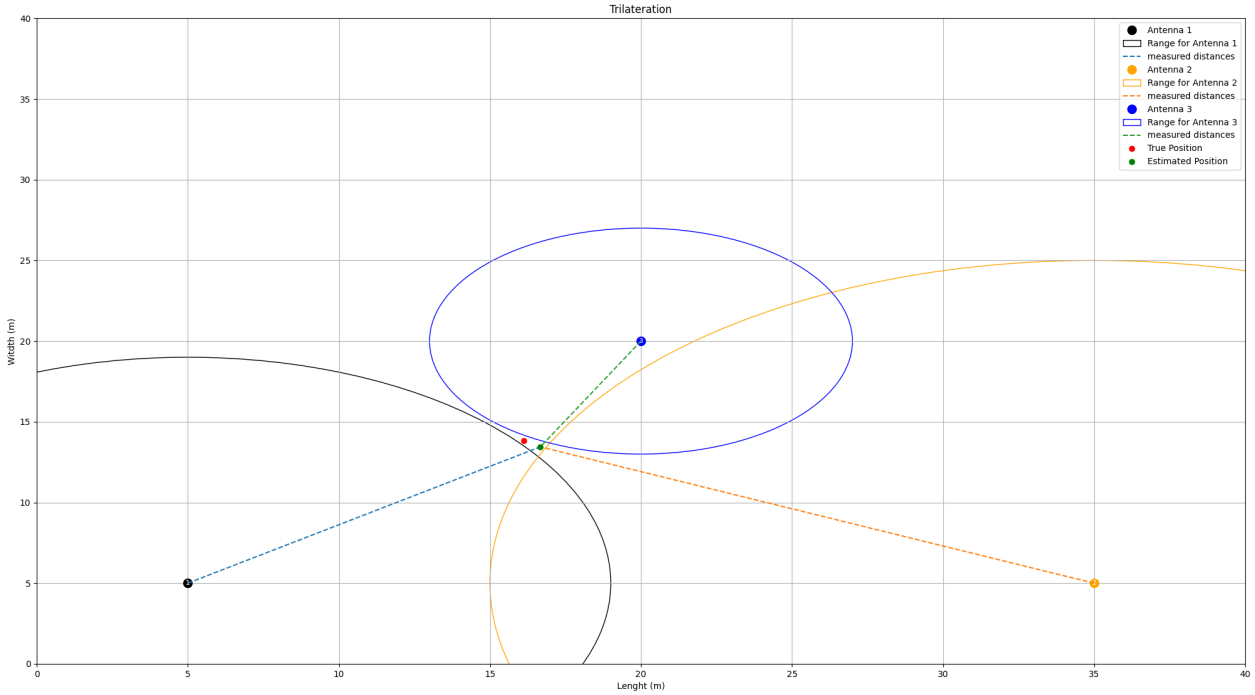


FIGURE 4 – Implémentation de la multilatération

5 Étude de l'optimisation de l'emplacement des antennes WIFI avec l'algorithme BPSA (Binary Particle Swarm Optimization)

5.1 Algorithme BPSA (Binary Particle Swarm Optimization)[3]

5.1.1 Dilution Of Precision (DOP)

Le Dilution Of Precision (DOP) est une mesure qui décrit l'effet de la géométrie des balises sur la précision de la localisation. Un DOP faible indique une géométrie favorable qui minimise les erreurs de localisation, tandis qu'un DOP élevé indique une géométrie moins optimale qui peut augmenter l'erreur. Le DOP est crucial dans l'optimisation de l'emplacement des antennes car il aide à évaluer la qualité d'une configuration d'antennes en termes de précision potentielle de localisation.

5.1.2 Paramètre d'évaluation RMSE

L'erreur quadratique moyenne de positionnement (RMSE) est un paramètre d'évaluation crucial pour la localisation. Il est défini comme la racine carrée de la moyenne des carrés des écarts entre les positions estimées et les positions réelles. Formellement, le RMSE pour un ensemble de localisations estimées est donné par:

$$RMSE(\Omega_n) = \sqrt{\frac{1}{L}\sum_{i=1}^{L}(\mathbf{x}_i - \hat{\mathbf{y}}_i)^2} \tag{5}$$

où Ω_n représente la configuration des balises trouvée par BPSA sous un critère DOP, $\mathbf{x}_i$ est la position réelle de la cible, $\hat{\mathbf{y}}_i$ est l'estimation de la position obtenue par trilatération, et L est le nombre total de localisations estimées.

5.1.3 Principe de l'algorithme BPSA[3]

L'algorithme BPSA est une méthode d'optimisation qui ajuste la position des balises pour minimiser le DOP et le RMSE, ce qui améliore la précision de la localisation. Le processus BPSA comprend les étapes suivantes:

1. Initialisation des positions des balises selon une distribution prédéfinie ou aléatoire dans l'espace de recherche.
2. Évaluation de la configuration des balises en utilisant le DOP et le calcul du RMSE pour la localisation estimée.
3. Application d'une stratégie d'optimisation pour ajuster les positions des balises afin de réduire le DOP et le RMSE.
4. Répétition des étapes d'évaluation et d'optimisation jusqu'à ce que la convergence soit atteinte ou que le nombre maximal d'itérations soit réalisé.

La finalité est de trouver une configuration des balises qui offre la meilleure géométrie pour une localisation précise en minimisant à la fois le DOP et le RMSE.

5.2 Optimisation du placement des antennes WIFI pour la simulation par BPSA

L'objectif de l'algorithme BPSA (Binary Particle Swarm Optimization) est de trouver l'emplacement optimal des antennes WiFi pour maximiser la couverture et la précision de la localisation en intérieur. Le Dilution Of Precision (DOP) est calculé en tenant compte de l'atténuation du signal WiFi, qui suit un modèle logarithmique de la puissance. De plus, des contraintes matérielles sont imposées, telles que le fait que les balises doivent être fixées au plafond, limitant ainsi l'espace de recherche des positions optimales.

5.2.1 DOP et atténuation des ondes WIFI[4]

Le Dilution Of Precision (DOP) est une mesure de la qualité géométrique d'un ensemble de points de mesure (antennes). Il est calculé en prenant en compte l'atténuation des ondes WiFi qui suit une loi de puissance logarithmique, exprimée par la formule suivante où $P(d)$ est la puissance reçue à une distance d, $P(d_0)$ est la puissance de référence à une distance d_0, et n est le coefficient d'atténuation.

$$P(d) = P(d_0) - 10n \log_{10}\left(\frac{d}{d_0}\right) \tag{6}$$

5.2.2 Paramètre d'évaluation RMSE

La Root Mean Square Error (RMSE) pour l'emplacement des antennes est calculée en utilisant les positions estimées et réelles:

$$RMSE(\Omega_n) = \sqrt{\frac{1}{L}\sum_{i=1}^{L}(\mathbf{x}_i - \hat{\mathbf{y}}_i)^2} \tag{7}$$

Ici, Ω_n représente la configuration des balises optimisées par BPSA, $\mathbf{x}_i$ est la position réelle, et $\hat{\mathbf{y}}_i$ est la position estimée.

5.2.3 Résultats

L'optimisation de l'emplacement des antennes WiFi en utilisant l'algorithme BPSA a mené à des améliorations significatives de la distribution de la force du signal dans l'environnement simulé. Les résultats montrent une augmentation progressive de la couverture moyenne en pourcentage de la force du signal avec l'ajout de balises supplémentaires. Avec trois balises, la couverture moyenne était de 53%, ce qui s'est amélioré à 60% avec quatre balises, et a atteint 63% avec cinq balises. Cette progression suggère que l'ajout de balises contribue à une meilleure uniformité de la couverture du signal, ce qui est essentiel pour la localisation précise et fiable dans des applications telles que la navigation intérieure et le suivi d'objets.

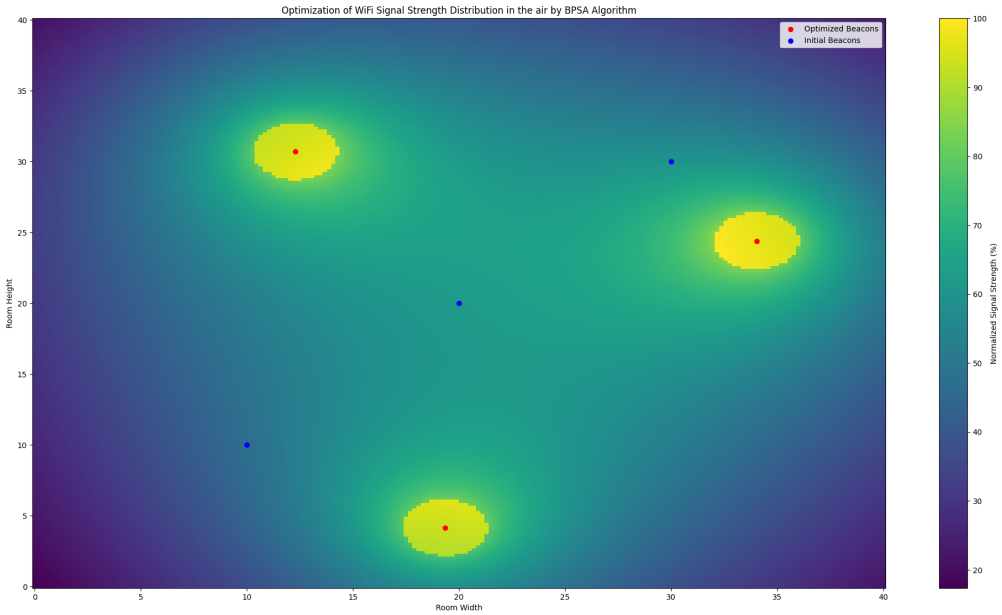


FIGURE 5 – Optimisation du placement par BPSA pour 3 balises

Cependant, il est important de noter que l'augmentation de la couverture diminue avec chaque balise ajoutée, indiquant des rendements décroissants. Cela implique qu'il existe un nombre optimal de balises au-delà duquel les améliorations de couverture deviennent marginales par rapport au coût d'installation supplémentaire.

En conclusion, l'algorithme BPSA se révèle être un outil efficace pour l'optimisation de la localisation des antennes WiFi, permettant d'atteindre un équilibre entre une couverture étendue et la minimisation des coûts. Les résultats soulignent l'importance d'une planification stratégique dans le déploiement des réseaux sans fil pour des applications de localisation intérieure, avec un nombre de balises optimisé pour une performance maximale.

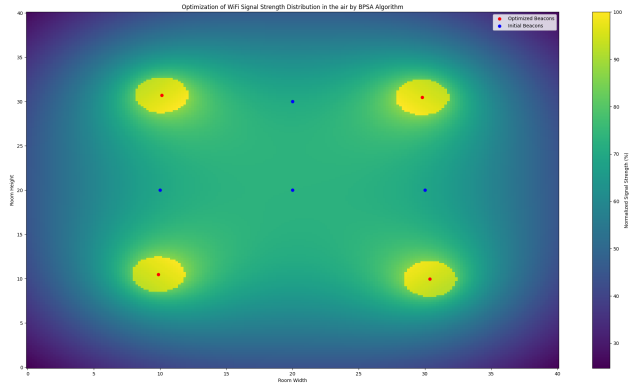


FIGURE 6 – Optimisation du placement par BPSA pour 4 balises

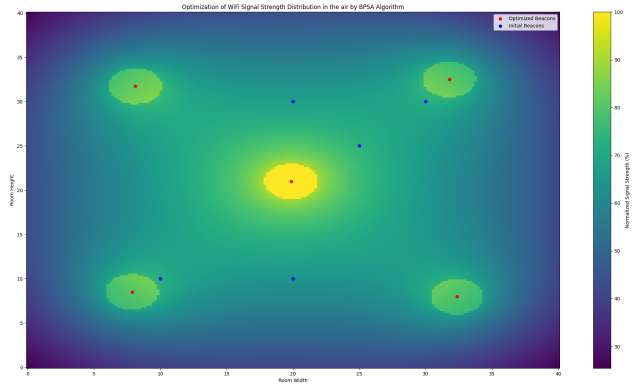


FIGURE 7 – Optimisation du placement par BPSA pour 5 balises

6 Localisation du robot par ondes WIFI

6.1 Simulation avec les obstacles et les 9 antennes

La simulation de la localisation du robot par ondes WiFi a été réalisée dans un espace 3D configuré avec 9 antennes WiFi et des obstacles modélisés pour émuler un environnement intérieur complexe. Les antennes ont été placées stratégiquement pour maximiser la couverture du signal et minimiser les interférences et les zones d'ombre causées par les obstacles. La position estimée du robot est obtenue par l'application de l'algorithme de trilatération, qui utilise la puissance des signaux reçus des antennes pour déterminer sa localisation.

Les figures 8 et 9 présentent deux vues différentes de la simulation. Les points bleus représentent les antennes, tandis que le point rouge illustre la position réelle du robot, et le point vert indique la position estimée obtenue par la simulation. Les rectangles rouges représentent les obstacles dans l'espace, tels que les murs ou le mobilier, qui peuvent affecter la propagation des signaux.

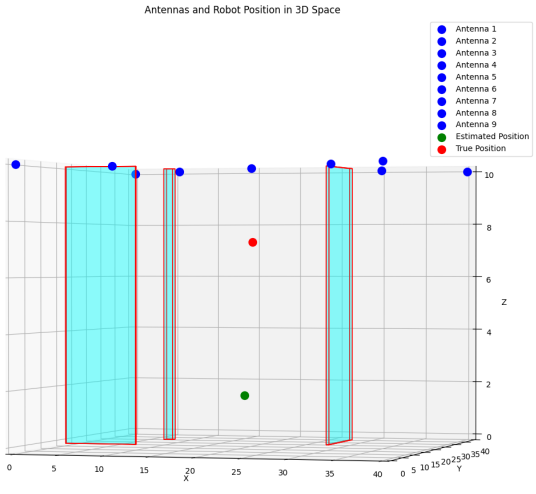


FIGURE 8 – simulation 3D vue 1

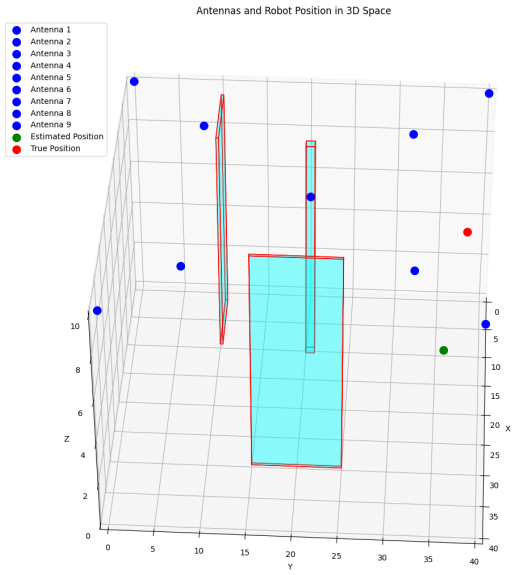


FIGURE 9 – simulation 3D vue 2

6.2 Étude de l'influence du nombre de balises sur l'erreur de localisation

Les erreurs de positionnement sont plus faibles avec trois balises (0.61 dans l'air et 1.07m avec obstacle). Cependant, au-delà de quatre balises, on observe une décroissance des moyennes d'erreurs de position qui semble converger. La précision accrue avec trois balises peut être due à la manière dont l'erreur sur la position est générée (ajout d'un bruit proportionnel à la puissance du signal). Une manière de pallier ce problème pourrait être de réaliser la trilatération uniquement avec les trois balises les plus proches.

Dans les environnements sans obstacles, l'amélioration de la précision est plus marquée, probablement en raison de la propagation directe des signaux. Les obstacles, tels que les murs, introduisent des réflexions et des atténuations du signal qui perturbent la mesure de la distance, entraînant une réduction de la précision.

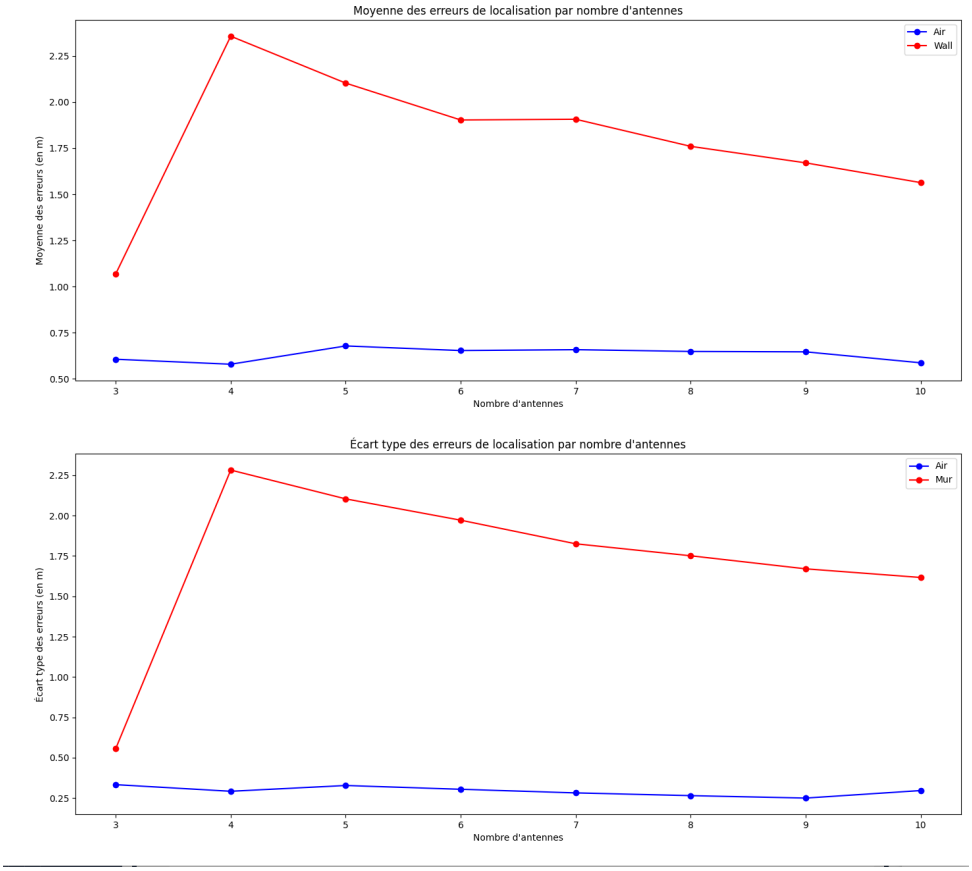


FIGURE 10 – Moyenne et écart type des erreurs de localisation en fonction du nombre de balises dans des environnements avec et sans obstacles.

Conclusion

À travers cet article, nous avons exploré l'état de l'art des technologies et des méthodes de localisation en intérieur, mettant en évidence les défis et les opportunités inhérents à ce domaine en évolution rapide. L'optimisation de l'emplacement des antennes WiFi via l'algorithme BPSA illustre l'importance de l'application de techniques d'optimisation avancées pour améliorer la précision et la fiabilité de la localisation intérieure. En tenant compte de l'atténuation des ondes et des contraintes matérielles, les résultats obtenus démontrent une amélioration substantielle de la distribution du signal et de la précision de localisation avec une augmentation du nombre de balises.

Cette recherche souligne la nécessité d'une planification minutieuse et d'une conception réfléchie lors du déploiement des systèmes de localisation intérieure, en tenant compte des caractéristiques uniques de chaque environnement. Alors que nous progressons vers un avenir de plus en plus connecté, les travaux présentés ici offrent des orientations précieuses pour la mise en œuvre de solutions de localisation intérieure robustes et efficaces, capables de répondre aux exigences des applications émergentes dans des domaines tels que l'automatisation industrielle, la santé connectée et la domotique intelligente.

Table des figures

Références

[1] Wajih Abdallah. *La résolution du déploiement 3D d'objets connectés sans fil à l'intérieur en utilisant un schéma hybride entre les méthodes géométriques de déploiement et les algorithmes d'optimisation distribués*. Ph.d. thesis, Université Toulouse le Mirail - Toulouse II, 2022. NNT : 2022TOU20043. ii, 4

[2] Jean-Pierre Barbot, Isabelle Kyoko Vin, Pan Liu, Ludovic Chamoin, and Dominique Placko. Géolocalisation et navigation à l'intérieur des bâtiments. In *URSI France*. Laboratoire SATIE, ENS Paris-Saclay, 2022. ii, 1, 4, 7

[3] Soufien Kammoun. *Géolocalisation à l'intérieur d'un bâtiment pour terminaux mobiles*. Ph.d. thesis, Télécom ParisTech, 2016. NNT : 2016ENST0041. ii, 1, 4, 9

[4] Frédéric Lassabe. *Géolocalisation et prédiction dans les réseaux Wi-Fi en intérieur*. Ph.d. thesis, Université de Franche-Comté, 2009. NNT : 2009BESA2062. ii, 2, 4, 5, 10

Deterministic model comparison for reinforcement learning in a robotic arm environment.

Taddeo Guérin

KEYWORDS

Reinforcement Learning, Robotic Arm, Deep Learning, Agent-Critic, DDPG, TD3, panda-gym.

ABSTRACT

This article presents different AI models in a reinforcement learning context. We compare the AC (Agent-Critic) reinforcement learning model with the deep reinforcement learning models DDPG (Deep Deterministic Policy Gradient) and TD3 (Twin-Delayed Deep Deterministic Policy), in the context of a robotic arm. To do this, we use panda-gym's Panda Reach environment, which simulates the Franka Emika arm by adding the functions required for reinforcement learning. The various models must learn to move the robot's gripper to a specific target, and are rewarded accordingly. Finally, this article shows that the deep models (DDPG and TD3) are much faster at learning this task, learning twice as fast as their shallow counterpart (AC).

INTRODUCTION

Rapid advances in the field of robotics have transformed the way machines interact with their environment, offering new and promising prospects for improving various sectors of society. Among the most striking innovations, robotic arms are emerging as a crucial component, redefining the manipulation and interaction capabilities of machines. These devices, designed to mimic human dexterity, represent the pinnacle of convergence between advanced mechanics and artificial intelligence. Within this synergy, reinforcement learning stands out as an automated learning methodology that has revolutionized the way robots assimilate and improve their skills over time. This approach is inspired by the learning paradigm based on rewards and punishments, offering machines the ability to optimize their actions according to the consequences of their decisions. In the context of robotic arms, reinforcement learning enables robots to adapt autonomously to dynamic and complex environments. In recent years, numerous studies have applied deep reinforcement learning algorithms to the manipulation of robotic arms, with excellent results.

This article explains the basics of deep reinforcement learning, as well as the results of implementing deterministic Actor-Critic, DDPG and Twin-Delayed models in the panda-gym library environment.

1 PRELIMINARY KNOWLEDGE

1.1 Reinforcement Learning

Reinforcement learning is a complete, interactive, goal-oriented agent that interacts with the environment and learns what to do to maximize reward. It belongs to a branch of machine learning, but differs from supervised and unsupervised learning, which are widely used in the field of machine learning. Supervised learning aims to derive prediction functions from labeled training data. However, in reinforcement learning, the agent cannot obtain enough labeled actions. Unsupervised learning aims to find hidden structures from unlabeled training data. Reinforcement learning aims to obtain the greatest reward, not the structure of the data. The Markov Decision Process (MDP) represents the mathematically ideal form of reinforcement learning problems. In most cases, the algorithm's performance can only be guaranteed if the reinforcement learning problem can be abstracted in MDP form. In the reinforcement learning system, there are four main sub-elements: a policy, a reward signal, a value function and possibly a model of the environment. Figure 1 shows the "agent-environment" interaction of the Markov decision process.(1)

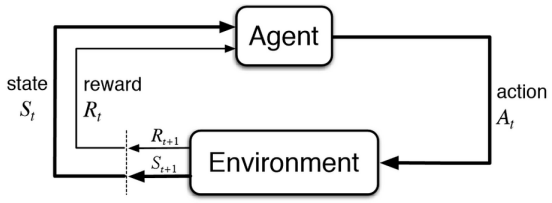


Figure 1: Agent-environment interaction

MDP is as tuple : MDP $=\big(S,\quad A,\quad P,\quad R,\quad \gamma\big)$ where :

- S: The set of states of the environment that the robot interacts with $s_t \in$ S is the state of the Agent at time t;
- A: The set of executable robot actions, $a_t \in$ A is the action performed by the robot at time t, such as joint rotation, back-and-forth movements, etc;
- P_A: The dynamic/transition model for each action;
- R: a reward function;
- $\gamma \in$ [0,1]: the discount factor.

Finally, the objective of this learning process is to find the optimal value function v* and the optimal policy π*.(1)

1.2 Deep Learning

Deep neural networks, or artificial neural networks, seek to mimic the human brain through a combination of data inputs, weights and biases. These elements work together to accurately recognize, classify and describe objects within the data. Deep neural networks consist of several layers of interconnected nodes, each building on the previous layer to refine and optimize prediction or categorization. This progression of computations through the network is called forward propagation. The input and output layers of a deep neural network are called visible layers. The input layer is where the deep learning model ingests data for processing, and the output layer is where the final prediction or classification is made. Another process called backpropagation uses algorithms, such as gradient descent, to calculate prediction errors, then adjusts the weights and biases of the function by moving backwards through the layers in order to train the model. Together, forward propagation and backpropagation enable a neural network to make predictions and correct errors accordingly. Over time, the algorithm becomes progressively more accurate. The algorithms I'm going to use belong to this branch of machine learning.(2)

2 THE ENVIRONMENT: PANDA REACH, FROM PANDA-GYM

I used the PandaReach-v3 environment, created using the PyBullet physics engine, which has the advantage of being open-source and offers very robust simulation performance. The environment is integrated with OpenAI Gym, enabling the use of all learning algorithms based on this programming interface. It simulates Franka Emika1's Panda robotic arm, a 7-degree-of-freedom arm with parallel finger grippers, as well as a target. The agent's task is to place the arm's gripper on the target and stay there. The reward is calculated according to the number of steps required to reach the target.(3)

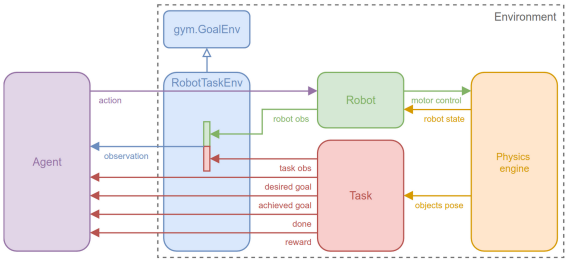


Figure 2: Simulation software architecture

The physic engine creates the arm and target in the environment, then sends their position back to the *Robot* topic and the *Task* topic. These topics then transmit their observations, so *Task* gives the agent the desired task and its reward. This information then leaves the environment to enter the agent's model, which will then give the command for an action to the Robot (in its topic), which will carry it out. The main class, called *RobotTaskEnv*, contains a robot attribute and a task attribute. When the agent performs an action and sends it to the environment, this action is transferred to the robot. The collected observation is the concatenation of robot-specific observations (such as the position of the gripper, for example) and task-specific observations (such as the position of objects, for example). Finally, to follow the Multi-Goal framework, the desired goal and the achieved goal are derived from the task attribute.(3)

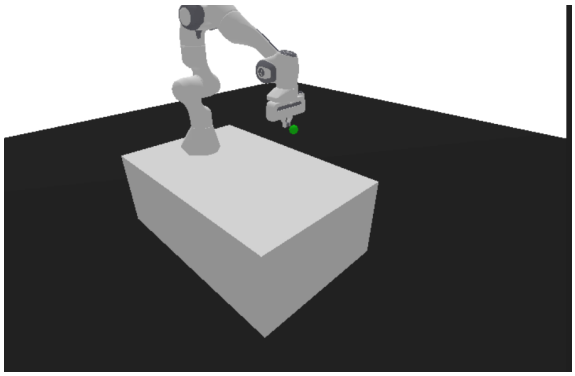


Figure 3: Panda Reach environment

3 REINFORCEMENT LEARNING ALGORITHMS

3.1 Agent-Critic

The actor-critic approach is a popular technique in RL that combines the advantages of value-based methods (such as Q-learning) and policy-based methods (such as policy gradient methods). In an actor-critic system, we have two main components:

- Actor: The actor is responsible for selecting actions. He learns a policy $\pi(a|s)$ that directly maps states to actions. Unlike value-based methods, the actor does not aim to estimate the value of each action, but rather to generate actions directly.

- Critic: The critic is responsible for evaluating the actions taken by the actor. It learns a value function $V(s)$ or $Q(s,a)$ that estimates the value of a given state or action. This estimate is used to evaluate the actor's performance and guide his learning

The key to the actor-critic approach lies in the way actor and critic interact and improve each other. Here's how it usually works:

1. The actor selects an action based on the current policy $\pi(a|s)$.

2. The action is executed in the environment.

3. The critic evaluates the quality of the action according to the value function $V(s)$ or $Q(s, a)$.

4. The error between the predicted value and the actual value is calculated and used to update the critic.

5. The actor is updated using the critic's error signal to improve its policy.(4)

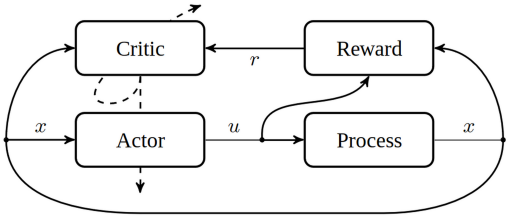


Figure 4: Agent-Critic learning

This feedback loop enables the actor to learn to select actions that maximize the value estimated by the critic. In return, the critic benefits from the actor by providing examples of real actions and helping him refine his value estimates. (4)

There are several variants of the actor-critic approach, but here I've decided to use a deterministic model. Here, the actor produces specific actions according to the observed state, rather than calculating a probability distribution as in a stochastic model. These deterministic models, often called DPG (deterministic policy gradient), are particularly useful in environments where actions need to be precise and deterministic, for example in control tasks where precise actions are required to manipulate a physical system. However, they can lack computing power, which is why the Deep deterministic policy gradient was created.

3.2 DDPG

The exact algorithm can be seen below:(5)

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta_Q)$ and actor $\mu(s|\theta_\mu)$ with weights θ_Q and θ_μ.
Initialize target network Q' and μ' with weights $\theta_{Q'} \leftarrow \theta_Q, \theta_{\mu'} \leftarrow \theta_\mu$.
Initialize replay buffer R
for $episode = 1, M$ **do**
  Initialize a random process N for action exploration
  Receive initial observation state s_1
  for $t = 1, T$ **do**
    Select action $a_t = \mu(s_t|\theta_\mu) + N_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta_{\mu'})|\theta_{Q'})$
    Update critic by minimizing the loss:
    $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta_Q))^2$
    Update the actor policy using the sampled policy gradient:
    $\nabla_{\theta_\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta_Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta_\mu}\mu(s|\theta_\mu)|_{s_i}$
    Update the target networks:
    $\theta_{Q'} \leftarrow \tau\theta_Q + (1-\tau)\theta_{Q'}$
    $\theta_{\mu'} \leftarrow \tau\theta_\mu + (1-\tau)\theta_{\mu'}$
  end for
end for

Here's how the DDPG works:

1. Actor: The actor in the DDPG is responsible for selecting actions in a deterministic way. It takes as input the current state of the environment and directly produces an action. Unlike a stochastic actor, which produces a distribution of actions, the DDPG actor generates a specific action.

2. Critic (Critic): The critic in the DDPG is a value function that evaluates the quality of the actions taken by the actor. It takes the state and the action as inputs and estimates the value of this state-action pair. This makes it possible to evaluate the actor's performance and guide his learning.

3. Batch Learning: Stores transitions in a replay buffer and randomly samples mini-batches of these transitions for learning.

4. Actor and critic update: The actor and critic are updated using policy gradients and value gradients. The actor is updated using a deterministic policy gradient, i.e. the gradient of the value predicted by the critic with respect to the actor's parameters. The critic is updated by minimizing the loss between the estimated value and the target value, calculated using the actual reward and the next-state value predicted by the target critic.

5. Target Networks: To stabilize learning, the DDPG uses target networks to estimate the value of the next state in the critic's update. These target networks are updated slowly from the actor's and critic's main networks, enabling greater convergence.(5)

The DDPG can be further enhanced, in a variant called TD3.

3.3 Twin-Delayed Deep Deterministic Policy

The Twin Delayed Deep Deterministic Policy Gradient (TD3) is an extension of the Deep Deterministic Policy Gradient (DDPG), itself a variant of the Actor-Critic (AC) algorithm in the field of reinforcement learning (RL). TD3 aims to improve the stability and performance of DDPG by introducing several innovations. Firstly, TD3 uses two critics to estimate the action value. This configuration reduces the overestimation of action values by critics, a common problem in methods based on action-value functions (Q-functions). By taking the minimum value of both critics, TD3 mitigates the effects of overestimation errors. Secondly, TD3 uses a noisy target policy to explore the action space more efficiently. Unlike DDPG, which adds noise to the action selected by the main policy, TD3 adds noise to the target policy. This approach reduces the risk of overfitting to noise errors introduced by the main policy.(6)

In addition, TD3 uses a double-deterministic policy, meaning that it learns two deterministic policies. This further reduces the variance in action value estimates and contributes to better learning stability. Finally, TD3 uses a less frequent policy update than DDPG. This technique, known as delayed update, consists of updating the policy only every two critical updates. This stabilizes learning by reducing potential oscillations in action values and mitigating divergence problems.(6)

This algorithm can be seen here:(6)

Algorithm 2 TD3

Initialize critic networks Q_{θ_1}, Q_{θ_2}, and actor network π_ϕ with random parameters θ_1, θ_2, ϕ
Initialize target networks $\theta'_1 \leftarrow \theta_1$, $\theta'_2 \leftarrow \theta_2$, $\phi' \leftarrow \phi$
Initialize replay buffer B
for $t = 1$ to T **do**
  Select action with exploration noise $a \sim \pi_\phi(s) + \epsilon$, $\epsilon \sim N(0, \sigma)$ and observe reward r and new state s'
  Store transition tuple (s, a, r, s') in B
  Sample mini-batch of N transitions (s, a, r, s') from B
  $\tilde{a} \leftarrow \pi_{\phi'}(s') + \epsilon$, $\epsilon \sim \text{clip}(N(0, \tilde{\sigma}), -c, c)$
  $y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta'_i}(s', \tilde{a})$
  Update critics: $\theta_i \leftarrow \text{argmin}_{\theta_i} \frac{1}{N} \sum (y - Q_{\theta_i}(s, a))^2$
  if $t \mod d = 0$ **then**
    Update ϕ by the deterministic policy gradient:
    $\nabla_\phi J(\phi) = \frac{1}{N} \sum \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$
    Update target networks:
    $\theta'_i \leftarrow \tau\theta_i + (1 - \tau)\theta'_i$
    $\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$
  end if
end for

4 RESULTS

I therefore created three reinforcement learning algorithms: a deterministic actor-critic, a DDPG and a TD3. In order to be able to compare these models effectively, I kept the same hyperparameters, whether in terms of actors, critics or exploration factors. The training is carried out on the PandaReach-v3 simulation, which lets an articulated arm touch a target in space. I ran my tests on 300 episodes, with 50 steps per episode.

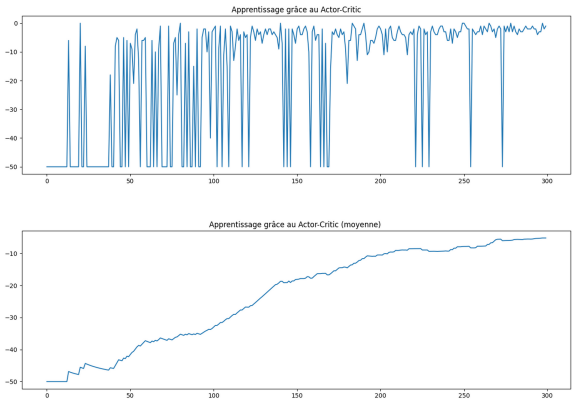


Figure 5: Results for the Agent-Critic Model

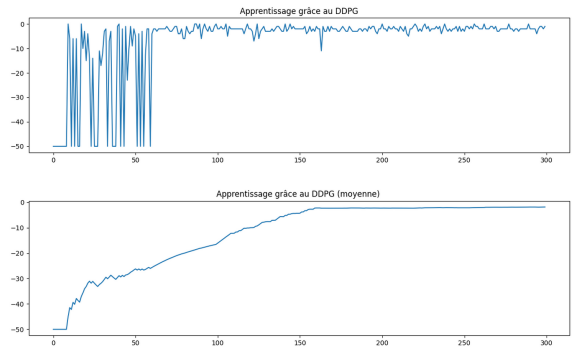


Figure 6: Results for the DDPG Model

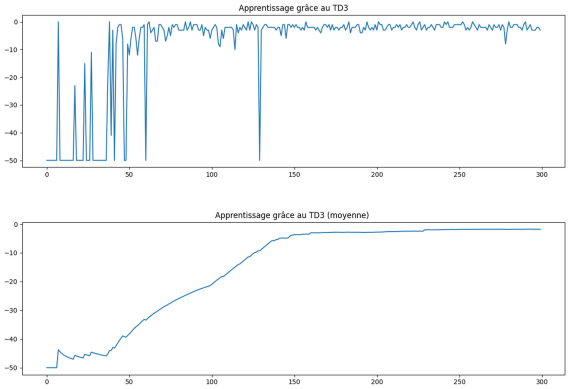


Figure 7: Results for the TD3 Model

As Figure 5 shows, the Agent-Critical model evolves relatively slowly. During the first 40 steps, it achieves very little of what is asked of it.Then comes a phase that might be called a transition phase. During this period, the model succeeds on average only half of the time. The model passes this phase in around sixty steps, before stabilizing. There are still times, however, when this model completely misses the mark, as can be seen from the downward-pointing peaks.

The DDPG model starts to converge much faster. From the 20th step onwards, it switches to transitional mode and remains in this mode for 40 steps, before converging. Compared with the previous model, this one is much faster and more efficient,with half as many steps in each phase, and a much higher success rate.

The TD3 model behaves in an interesting way. It stays longer than the DDPG model at a very low success rate (40 steps versus half for the DDPG). Nevertheless, it remains in the transition phase for a short time, with a success rate of 50%. While the other models spend 60 and 40 steps in this mode, the DDPG model spends only ten steps in it. In fact, it stabilizes extremely quickly towards the optimal model, enabling it to hit its targets quickly. This makes it all the more interesting.

5 CONCLUSION

In conclusion, we can see that all the models finally manage to perform the required task, but in a different number of steps. The DDPG, an improved model of Agent-Critic, is considerably faster than its predecessor, halving its learning speed. The TD3, on the other hand, closely resembles the DDPG. It finally stabilizes with the same number of steps as its predecessor, but with a different behavior: it stays longer in the failure phase, and very little time in the transition phase. The DDPG stays half as long in the failure phase, but twice as long in the transitional phase. This may be due to the complexity of the model, which takes longer to understand what's being asked of it, but gets there more quickly.

References

[1] H. Guan, "Analysis on Deep Reinforcement Learning in Industrial Robotic Arm," 2020 International Conference on Intelligent Computing and Human-Computer Interaction (ICHCI), Sanya, China, 2020, pp. 426-430.

[2] Ludovic Arnold, Sébastien Rebecchi, Sylvain Chevallier, Hélène Paugam-Moisy, "An Introduction to Deep Learning", European Symposium on Artificial Neural Networks (ESANN), Apr 2011, Bruges, Belgium.

[3] Quentin Gallouédec, Nicolas Cazin, Emmanuel Dellandréa and Liming Chen, "Multi-Goal Reinforcement Learning environments for simulated Franka Emika Panda robot", June, 2021

[4] Ivo Grondman, Lucian Busoniu, Gabriel Lopes, Robert Babuska. A survey of actor-critic reinforcement learning: standard and natural policy gradients. IEEE Transactions on Sys- tems, Man, and Cybernetics, Part C: Applications and Reviews, 2012, 42 (6), pp.1291-1307.

[5] Lillicrap, Timothy P, Hunt, Jonathan J, Pritzel, Alexander, Heess, Nicolas, Erez, Tom, Tassa, Yuval, Silver, David, and Wierstra, Daan. Continuous control with deep reinforcement learning. 2015.

[6] Dankwa, S., Zheng, W.: Twin-delayed DDPG: a deep reinforcement learning technique to model a continuous movement of an intelligent robot agent. In: Proceedings of the 3rd International Conference on Vision, Image and Signal Processing, pp. 1–5. Association for Computing Machinery, New York (2019).

Constrained Path planning with RRT*

Hugo Hofmann, Student from ENSTA Bretagne, Brest, France (e-mail: hugo.hofmann@ensta-bretagne.org)

***Abstract*—In the field of Robotics and particularly when it comes to autonomous robots, being able to plan paths or trajectories is usually very useful if not invaluable in the context of the mission the robots must accomplish. Many algorithms such as *Djikstra's* or the *A* algorithm*[1] have tackled the challenge of computing such paths within graphs with the main goal in mind being to find the shortest in terms of distance. However, because these methods revolve around a pre-existing graph, and in order to broaden the possibilities, this paper will explore the *RRT** algorithm and its use in state space path planning, by presenting an implementation and an attempt to integrate kinematic constraints to path planning.**

***Index Terms*—Path Planning, RRT, RRT*, Graph Search, Obstacle avoiding**

I. Introduction

In the dynamic realm of robotics, where autonomous agents navigate complex environments, the ability to plan efficient paths or trajectories is not merely advantageous but often essential for mission success. Traditional path planning algorithms such as Djikstra's and the A* algorithm have long served as pillars in the field, excelling at computing paths within predefined graphs with a primary focus on minimizing distance. However, as the demands of robotics expand into more intricate terrains and scenarios, the limitations of these conventional methods become increasingly evident.To broaden the spectrum of possibilities and cater to the evolving needs of autonomous systems, this paper delves into the realm of Rapidly-exploring Random Trees (RRT*) algorithm for path planning in state spaces. Unlike its predecessors, RRT* operates in a realm untethered by pre-existing graphs, offering a more adaptable approach to path computation. The flexibility of RRT* not only facilitates pathfinding in complex and dynamic environments but also opens avenues for the integration of diverse constraints, including kinematic considerations, into the planning process.

A. RRT* algorithm

Unlike other methods, the *RRT**[2] algorithm builds the graph it relies on as it goes, which allows for better planning. For instance, in the simple case of finding a valid path within a geographical 2D space with obstacles (ex: black pixels on an image), building the graph can allow to connect remote nodes together, which wouldn't be possible in 4 or 8-connected graphs where the paths found usually "zig-zag" and are rather unrealistic. Building the graph also allows for path planning in a non-discrete space, which opens the door to more accurate paths provided that the constraints can be properly and efficiently computed (ex: in the case of obstacles in a 2D space, 2D intervals can be used).

Algorithm 1 RRT* Algorithm

Require: Initial configuration q_{init}, Goal region Q_{goal}, Maximum number of iterations $K_{\max}$, Step size Δq, Neighborhood radius r, Collision checker function CollisionFree, Cost function c, and Parent pointer array Parent[.].
Ensure: Optimal path from q_{init} to Q_{goal}.

Initialize tree T with root node containing q_{init}.
for $k = 1$ to $K_{\max}$ **do**
  Generate random configuration q_{rand} in the configuration space.
  Find nearest node q_{nearest} in T to q_{rand}.
  Generate new node q_{new} by moving from q_{nearest} towards q_{rand} with step size Δq.
  if CollisionFree$(q_{\text{nearest}}, q_{\text{new}})$ **then**
    Find nearby nodes within radius r of q_{new}.
    Set $q_{\min} = q_{\text{nearest}}$.
    Set $c_{\min} = \infty$.
    for each nearby node q_{near} **do**
      if CollisionFree$(q_{\text{near}}, q_{\text{new}})$ **then**
        $c_{\text{near}} = c(q_{\text{near}}) + \text{dist}(q_{\text{near}}, q_{\text{new}})$
        if $c_{\text{near}} < c_{\min}$ **then**
          $q_{\min} = q_{\text{near}}$
          $c_{\min} = c_{\text{near}}$
        end if
      end if
    end for
    Set Parent$[q_{\text{new}}] = q_{\min}$.
    Add q_{new} to T.
    for each nearby node q_{near} **do**
      if CollisionFree$(q_{\text{new}}, q_{\text{near}})$ **then**
        $c_{\text{new}} = c(q_{\text{new}}) + \text{dist}(q_{\text{new}}, q_{\text{near}})$
        if $c_{\text{new}} < c(q_{\text{near}})$ **then**
          Parent$[q_{\text{near}}] = q_{\text{new}}$
        end if
      end if
    end for
  end if
end for
return Path from q_{init} to Q_{goal} using Parent[.].

II. Constrained Path Planning

A. Car kinematic Model

Let us consider a car of length L with position (x, y) in space and heading θ. Assuming the car does not slip, we can consider the linear speed v as well as acceleration and steering angle commands, respectively a and δ. We obtain a simple

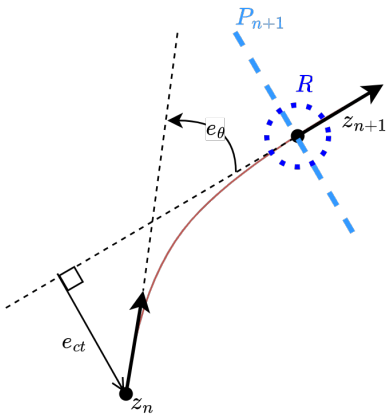


Fig. 1: Computing the path between two nodes z_n and z_{n+1}

kinematic model with state $X = [x, y, v, \theta]^T$.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{v} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ a \\ \frac{v\tan(\delta)}{L} \end{bmatrix} = f(X, u) \quad \text{with } u = [a, \delta]^T \tag{1}$$

This kinematic model is the most simple describing a car, and in further versions of the algorithm it could be improved, taking into account more dynamics-related constraints[3].

In order to do path planning in the state space, we also need to define the constraints associated with the space. In the case of the car, let's define maximum centripetal acceleration $a_{c_{max}}$ so that at all times $|a_c| < |a_{c_{max}}|$. This makes sure (under reasonable but of course uncomplete assumptions) that the vehicle does not slip. $a_{c_{max}}$ depends on the vehicle's characteristics. We can then link the state and commands of the system with this constraint by writing the curve radius R

$$R = L/\sin(\delta)$$

$$a_c = \frac{v^2}{R} = \frac{v^2\sin(\delta)}{L} \tag{2}$$

To properly apply the algorithm and especially the *steering* function, we need a control strategy which corresponds to the aim of the path planning. In the case where we want the car to reach its destination as quickly as possible, we can use the following strategy :

$$v_R = \min(v_{n+1}, \sqrt{\frac{a_{c_{max}} L}{\sin(\delta)}}) \tag{3}$$

$$a = clamp_{[-a_m, a_m]}(K_p(v_R - v)) \tag{4}$$

$$\delta = clamp_{[-\delta_m, \delta_m]}(K_{str}\arctan(K_{ct}e_{ct}) + K_{str}e_\theta) \tag{5}$$

The algorithm requires randomly generating nodes (ie, states); here the generation needs to account for the constraint on v, so that $X \in \mathbb{D}_x \times \mathbb{D}_y \times \mathbb{D}_v \times \mathbb{D}_\theta$ where $\mathbb{D}_x$ and $\mathbb{D}_y$ define the 2D boundaries of the environment, $\mathbb{D}_\theta = [0, 2\pi]$ and $\mathbb{D}_v = [0, \sqrt{\frac{a_{c_{max}} L}{\sin(\delta_m)}}]$.

Note that because of the non-holonomic nature of the car kinematic model, we cannot expect the robot to reach each node exactly, as there is no asymptotically stable continous control law that will park a non holonomic system at a point[4]. Instead, we need to use a criteria to quickly determine if a path is viable or not (apart from the collision from obstacles). As illustrated in Figure 1, we can define this criteria using the half-plane P_{n+1} defined by z_{n+1} and validation radius R:

1) Integrate the system using the control law until the vehicle crosses the limit P_{n+1}
2) The path is viable if the vehicle is in validation radius R around z_{n+1}

The R criteria can (and should) in fact be applied not just in the 2D geographical space (x, y) but also in the entire state space, so that the algorithm makes sure the robot can reach the node with the right speed and heading. This can be done by defining a norm in $\mathbb{R}^4$, or by splitting the criteria into several (one in the (x, y) plane, one with the speed error and one with the heading error).

III. Implementations

A. Shortest paths in euclidean distance

The first test of the algorithm is the most simple one, where the criteria to determine the path is the simplest, and instead of integrating the vehicle's kinematic model we simply consider straight lines between the nodes, using their length as the "cost" for each edge of the graph. Here the state variables are simply $[x, y]^T$. The whole program was written in Python using the *Numba* module, which pre-compiles functions when they're called for the first time and allows for faster computing times. In order to test the algorithm in various situations and for debugging purposes as well, an interface was developed allowing the user to draw obstacles, load images, etc.

Fig. 2: Path found on an urban map

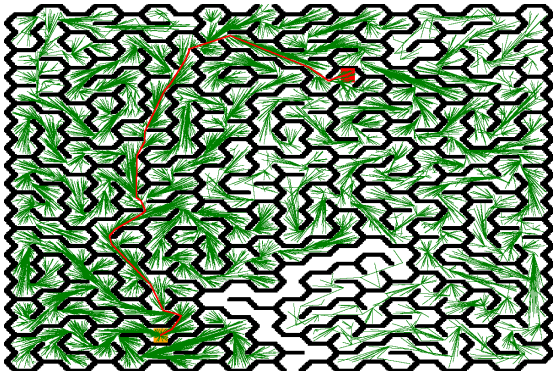


Fig. 3: Tree graph (green) and path (red) in an hexagonal maze

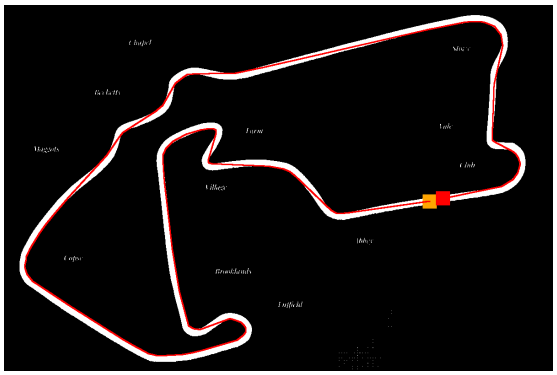


Fig. 4: Shortest (but obviously not fastest) trajectory on the Silverstone racetrack

Figure 3 shows how despite the fact that eventually the algorithm does converge towards the optimal solution as we add more and more nodes to the graph, the way the nodes are randomly generated can greatly influence performances when it comes to the number of iterations needed to cover the entire environment. We can see that with the lower area of the maze being empty of any links. On the other hand, we may also consider that if we only need to reach one goal, many of the nodes, generated in completely opposite directions of that of the destination, may be "useless": in that case we may want to try to speed up the search by generating the nodes "towards" the destination, in a similar fashion as the A* algorithm [1].

B. Constrained Path planning

In this section we implement the previously introduced kinematic model so that the algorithm can account for the way the vehicle moves. We could, just like before in III-A, compute the shortest path using the length of the path as the cost function. We obtain in this case:

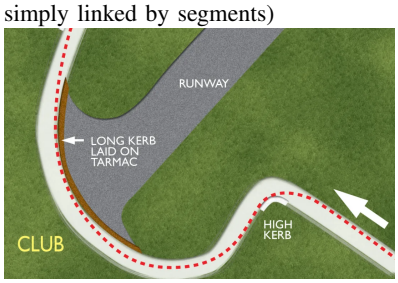
$$C(x,u) = \int_0^{t_{end}} v(t)dt \tag{6}$$

In reality, we are now free to use any function using trajectory $x(t)$ or commands $u(t)$ or a combination of both for the cost function. For example, let us say that we now want to find the fastest trajectory to the destination. The corresponding cost function is now :

$$C(x,u) = t_{end} \tag{7}$$

where t_{end} is the total time of the trajectory.

(a) Simulation of the fastest path with 2114 nodes generated (the actual curves are not displayed here, and the nodes are simply linked by segments)



(b) Reference race line on the Silverstone racetrack [5]

Fig. 5: Attempting to find the fastest path on the Silverstone racetrack

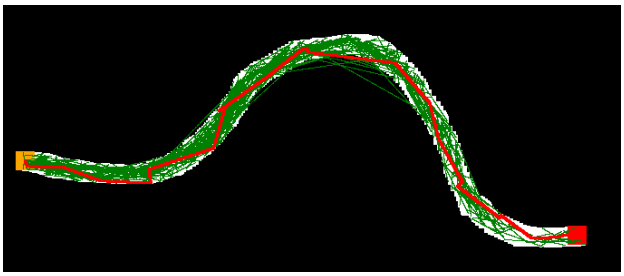


Fig. 6: First results from the constrained planning

The results show that the algorithm is almost functional; in Figure 5a we can see the trajectory being quite close to what one might expect from a usual race line on a racetrack. The major issue remains the computation time (here several minutes).

IV. Limitations, Improvements and Further Applications

A. Better node sampling

Applying the algorithm with the constraints presented earlier reveals the main issue with this method, which is generating "relevant" state nodes quickly and efficiently in the state space, which is particularly computationally inefficient considering the large number of computations for each potential link

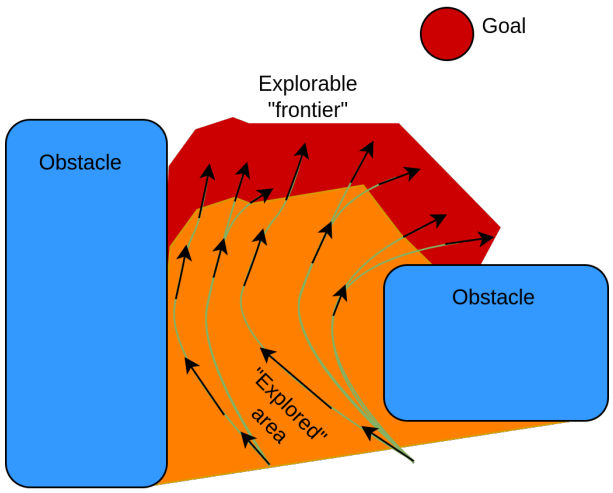


Fig. 7: Dynamic generation of nodes

between two nodes. One approach to improve on this issue would be to think of a better way to generate the nodes.

Although not implemented or tested, a method is proposed here: an algorithm of "exploration" that takes into account the previously generated nodes (in the (x, y) plane if not in the entire state space) to split the state space into three areas as illustrated in Figure 7:

1) *Explored* area: the area enclosing or at least representing all previously generated nodes. Can be computed through several ways (ex: convex hull of the node set).
2) *Explorable frontier*: the area at the edge of the *Explored* area; prefered area to generate new nodes.
3) The rest: obstacles, empty areas, etc

This method allows for efficient generation of nodes, by using a strategy similar to the ϵ-greedy policy one may use in the field of reinforcement learning[6]. We may thus define exploration parameter $\epsilon \in]0, 1[$ to use in the new sampling algorithm 2.

Algorithm 2 Rapid Exploration Sampling Algorithm (RESA)

$n \leftarrow random(0, 1)$
if $n \geq \epsilon$ **then**
  $z_{new} \leftarrow sample(explored\ area)$
else
  $z_{new} \leftarrow sample(frontier)$
end if
return z_{new}

This method should avoid generating too many nodes that end up having no valid parent within the graph,

B. Application in complex manoeuvers

Using kinematic constraints, the RRT* algorithm is able to compute complex behaviors. By changing the method a little bit, for example, by changing the state space the nodes are generated in, and adding different constraints, one may apply the method for different applications, including one of non continuous paths. In particular, in the case of a car-like robot, we may be interested in adapting the algorithm so that it yields a path giving the optimal strategy for a complex manoeuver such as a parking exercise. In this case each node would represent a stopping point (hence $\forall n$, speed $v_n = 0$). The constraints on the speed would also be updated to allow for negative speed in between nodes.

C. Tire wear constraint

Far from this paper's author the idea that speed is not a great (and fun) criteria when computing paths. However one may sometimes need to account for other aspects, including the longevity. Besides the autonomy, we may for example be interested in sparing the tires. Using a simple tire fatigue model such as Schallamach's[7], we may be able to compute paths that tend to minimize the tire fatigue over long periods of time, an interesting perspective in the field of industrial robotics for instance.

V. Conclusion

The developed simulation does seem to prove that the concept works, with satisfying paths being found based on several constraints set. However, besides many small improvements and further experiments, one major issue remains in the case of this implementation, which is the computational efficiency of the algorithm, and especially the sampling part.

References

[1] W. Zeng and R. L. Church. Finding shortest paths on real road networks: The case for A*. April 2009. https://zenodo.org/records/979689.

[2] Sertac Karaman, Matthew R. Walter, Alejandro Perez, Emilio Frazzoli, and Seth Teller. Anytime Motion Planning using the RRT*. In *2011 IEEE International Conference on Robotics and Automation*, pages 1478–1483, Shanghai, China, May 2011. IEEE. http://ieeexplore.ieee.org/document/5980479/.

[3] R. Frezza, A. Beghi, and G. Notarstefano. Almost Kinematic Reducibility of a Car Model with Small Lateral Slip Angle for Control Design. In *Proceedings of the IEEE International Symposium on Industrial Electronics, 2005. ISIE 2005.*, pages 343–348, Dubrovnik, Croatia, 2005. IEEE. http://ieeexplore.ieee.org/document/1528934/.

[4] R W Brockett. Asymptotic Stability And Feedback Stabilization.

[5] Circuit in detail - Pictures. https://www.evo.co.uk/advice/1952/circuit-in-detail-pictures.

[6] Richard S Sutton and Andrew G Barto. Reinforcement Learning: An Introduction.

[7] Marcus Grip. Tyre Performance Estimation during Normal Driving.

Utilisation de la couleur dans la reconnaissance d'objet en robotique marine

Introduction à la recherche

Emilie Ledoussal
Master Robotique
ENSTA Bretagne
Brest, France
emilie.ledoussal@ensta-bretagne.org

***Abstract*—This article looks at the use of color for object recognition in marine robotics. We'll refer here to several ways of getting around the problem of color attenuation due to water turbidity, assuming that the object can still be seen.**

Editor ***Index Terms*—robotic vision, color, marine robotics**

I. Introduction

Exploring the seabed is a major challenge for many people. Given the difficulty of sending directly a human being into this hostile environment, the advent of marine robotics has revolutionized this approach, using cameras and various sensors to provide interesting feedback, enabling us to map the seabed, monitor water quality, study marine life and much more.

In particular, let's look at the challenges posed by object recognition in an underwater environment. Low visibility is one of the most obvious and constraining. Unlike terrestrial environments where light can easily penetrate, underwater waters absorb and disperse light, reducing visibility and often making visual recognition difficult or impossible. In particular, colors can be altered, which can pose problems for overly simple image processing.

Color is an interesting feature in marine robotics. In particular, because of its robustness and ease of use. It can be used for detection and classification of underwater fauna, locating submerged objects and mapping the seabed. Despite all of this, underwater color vision is greatly complicated by the non-constant attenuation of vision, due, for example, to algae and sediments.

II. Theoretical notions

A. Light and color of an object

1) Colour perception of an object: The perception of an object's color is the result of various physical components, three of which are listed below. Firstly, the spectral composition of the light illuminating the colored object. The same object will have a different color if illuminated by the sun or an UV lamp. Secondly, the spectral reflectance of the object, i.e. its light-reflecting properties as a function of its composition and surface. Finally, the transmission of light through the surrounding medium, such as air or water. It's this third component in particular that we're interested in here, since it's through water that we lose information about the object's color.

2) Object Lighting: Here, we're going to look at the lighting of an object. When an object is illuminated, it absorbs some of the light and reflects the rest. Reflection depends on various parameters, such as :

- the position and direction of the light source
- the color and intensity of the light source
- the position and direction of the object
- the object's absorption properties
- the position and direction of the observer.

Two types of reflection must be taken into account among these parameters: diffuse reflection and specular reflection. Diffuse reflection occurs on rough surfaces, scattering light in many directions, while specular reflection occurs on smooth surfaces, reflecting light at a precise angle determined by the law of reflection.

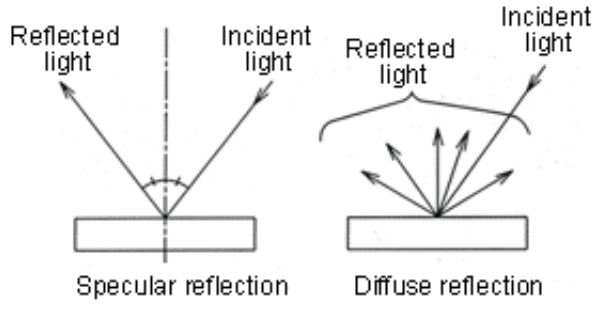


Fig. 1. Diagram illustrating the difference between diffused and specular light

Let's assume that the subject of this study is an unused Riptide robot shell, used as a colored marker in several ENSTA Bretagne projects (this target is red/orange). Its surface is

smooth, which suggests that we could focus on specular reflection.

Fig. 2. Color out of the water of the Riptide Shell used as an example

3) Colours features: In this paragraph, we'll look at how to characterize color for this subject. There are several ways of characterizing color, the best known probably being RGB, whose model is based on the fact that each color is represented by a mixture of three basic components: red, green and blue. Each component is generally expressed on a scale of 0 to 255, where 0 represents the absence of that component and 255 its full intensity.

A color can also be defined by three main parameters: hue, saturation and brightness.

- Hue refers to the type of color, such as red, blue or yellow, which depends on the dominant wavelength of light. For example, the hue of red is associated with a specific wavelength in the light spectrum (around 700 nm).
- Saturation, or purity, indicates the intensity or richness of a color. A pure color has a narrow spectrum, while an impure color, formed by a mixture of wavelengths, has a broader spectrum. In simple terms, a lower saturation gives a duller or more subdued appearance to the color.
- Brightness, also known as intensity, or value, measures the overall energy of the color spectrum. A brighter color appears brighter and more vivid due to its greater amount of energy.

This is the HSV model. In the next figure you can see a diagram representing both quite relevantly.

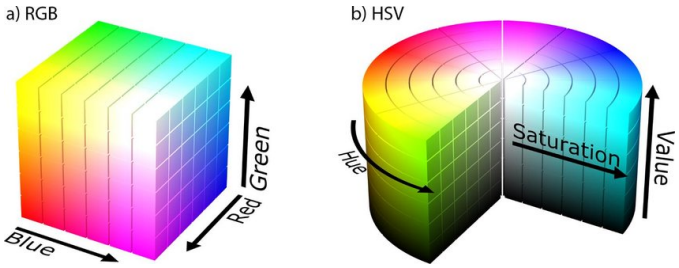


Fig. 3. RBG vs HSV[4]

B. Loi de Beer-Lambert

Beer-Lambert's law, a fundamental principle of spectroscopy, is an empirical law describes the absorption of light by a solution as a function of the concentration of this solution and the length of the optical path through which the light passes. In simple terms, this law states that the absorbance of a solution is directly proportional to the concentration of the absorbing substance in the solution, as well as to the thickness of the solution through which the light passes.Even if the Beer-Lambert law is usually used in chemistry, it can be applied in the hydrology field [5]. Here, the "solution" is the water in which the target and the robot filming it are immersed.

This law can be written as:

$$I_{\lambda,d} = I_{\lambda,0}.e^{-c_\lambda.d} \tag{1}$$

with λ as the wavelength, $I_{\lambda,d}$ as the observed intensity of light of corresponding λ at the distance d. $I_{\lambda,0}$ is the intensity of the light at the source (with a distance of 0), c_λ is the beam attenuation coefficient for the wavelength λ. c is a coefficient used in the oceanographic community called the beam attenuation coefficient at wavelength λ or total attenuation.

C. Compatibles colors

In this section we will define the concept of compatible colors introduced in the paper of S. Bazeille and I. Quidu and L. Jaulin [1]

We are going to draw directly on the demonstrations already carried out in the above-mentioned publication.

First, let's consider the definition given:

Definition: When underwater, considering the same object of a certain color illuminated by a natural or artificial light source, we define as compatible color all the colors perceived by changing the distance and lighting.

We need to determine how we are going to numerically recognize that colors are compatible.

First, we consider the vector $C = (C_R, C_G, C_B)$ which is the attenuation coefficients, we need to estimate them with the least squares method, we'll describe in a latter part.

For two colors y and $\overline{y}$ to be considered compatibles, they need to validate the following equality:

$\Psi(y,\overline{y}) \in [C-\epsilon, C+\epsilon]$ with $\Psi(y,\overline{y}) = \frac{log(y_G)-log(\overline{y}_G)-log(y_R)+log(\overline{y}_R)}{log(y_R)-log(\overline{y}_R)-log(y_B)+log(\overline{y}_B)}$ and $C = \frac{C_R-C_G}{C_R-C_B}$ with $C_R \neq C_B$

D. Use of Artificial Intelligence

Artificial Intelligence has been on the rise lately, enabling us to work around problems with sufficient data. With a data set based on underwater object recognition, and a CNN neural network, for example. It is then possible to determine the coefficients needed to improve the color of the image. Several researchers have been working on this approach, for example we can cite [6] among plenty of other articles on the subject.

III. Methods

In this section, we'll mention the methods we'll be using to: capture the images we'll be using next, color characterization, calculation of attenuation coefficients, and the subsequent processing to determine whether a color is compatible.

A. Image capture

To obtain the images used in this article, we used a BlueROV dedicated to us for the Guerledan EU. We took advantage of one of the weeks spent in Guerledan to obtain camera images of the target. The water in Guerledan is quite cloudy, so color fading is quite strong, which presents an interesting challenge for this article.

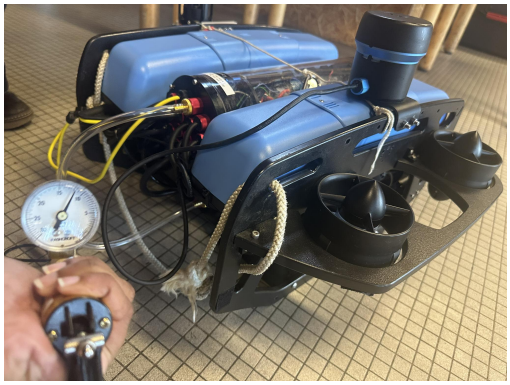


Fig. 4. Blinky, the BlueROV used to get the images in the Guerledan lake

As said earlier, the target here is a shell of a prototype of the Riptide Robot from ENSTA Bretagne, it is hanged on a buoy. The target is easily lost from sight (pretty much like the Riptide itself).

B. Characterization of target color out of and underwater

Let's take for example a few captures of the same target at a few distances. We'll take the colors from them with a color picker d0 being the color of the object out of the water. d1, d2 and d3 are unknown, because we don't have the exact distance, but we know for sure that $d0 < d1 < d2 < d3$

In figure 6 are the reference images used in the rest of the paper, and in figure 7, the color picked to represent each distance, to simplify the calculation. We can notice by reading the RGB values that the red component is being muted the bigger the distance, while the green and the blue components are increasing.

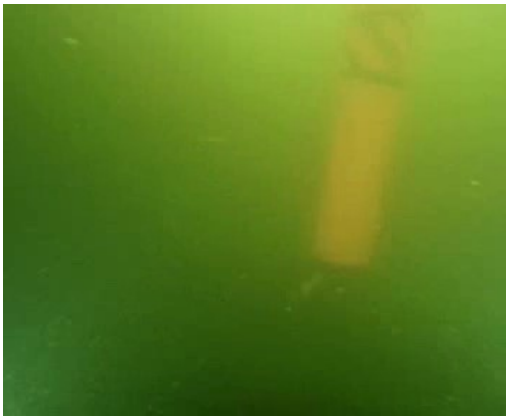


Fig. 5. Underwater target at Guerledan lake, filmed by Blinky

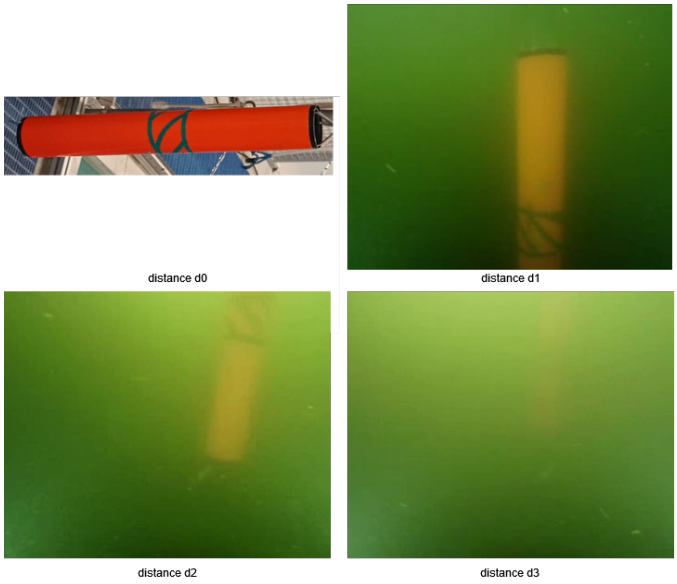


Fig. 6. Different pictures of the target, respectively at distance d0,d1,d2,d3

distance	HEX	RGB	HSV
d0	#D42E08	(212,46,8)	11°/96°/83°
d1	#A67B10	(166,123,16)	43°/90°/65°
d2	#A59829	(165,152,41)	54°/75°/65°
d3	#869835	(134,152,53)	71°/65°/60°

TABLE I
HEXADECIMAL CODE, RGB VALUE AND HSV VALUE FOR EACH DISTANCE D0,D1,D2,D3

C. Estimation of the attenuation parameters

Calculation of $C = (C_R, C_G, C_B)$

We write a python script that takes as input rgb triplets corresponding to the object's colors at different distances. The values are then normalized by dividing the components by the sum of the RGB values. Finally, we apply the method of least

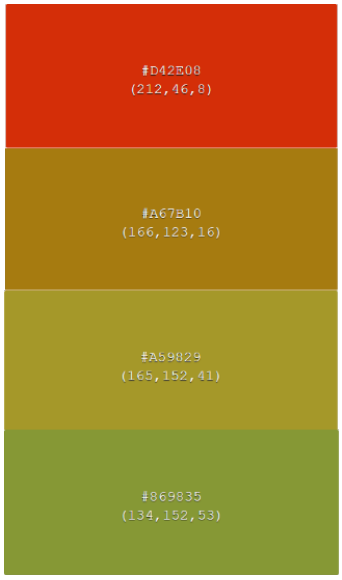


Fig. 7. Color of the target picked from the pictures

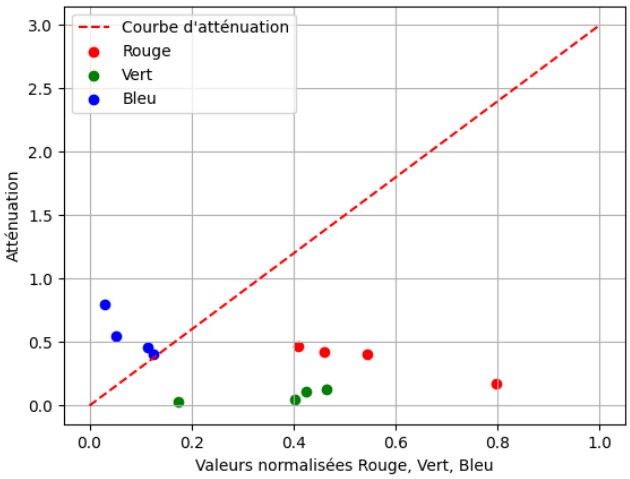


Fig. 8. Normalized RGB Value and approximation of the attenuation vector C

squares to obtain the attenuation curve based on the RGB values used.

D. Calculation of Ψ and the value C

We write a python script that takes as input the RGB components of a y color and a $\overline{y}$ color. This script applies the formula and returns a Ψ

In the same script, we can calculate the value C depending on the values of the vector C.

E. Image Processing

For practical image processing, we'd like to write a program that fragments the image into sub-images, processes them with a Gaussian filter, and extracts RGB triplets at several distances. We'll then calculate the C and Ψ vectors, and compare them to see if the colors are compatible. If so, we can consider image processing to 'repair' the color.

IV. Results

A. Calculation of $C = (C_R, C_G, C_B)$

We've tried to apply the calculation protocol for obtaining the C vector in a python script with the RGB values color picked-earlier, but the result we've obtained seems questionable, especially the curve, which doesn't seem to have adjusted to the normalized values (see figure 8). We'll continue using the average of the calculated attenuation values (which is also a questionable choice). So, we'll consider C=(0.55,0.36,0.08)

B. Calculation of Ψ and the value C

We have applied these calculations to the colors corresponding to distances d1 and d2 (d0 is a little too far away from the other values). We obtain Ψ=-0.229 and C=0.404, depending on the ϵ we choose, this result can seems satisfying.

Discussion

The whole program from the image to the corrected image hasn't been written, but we at least validated the notion of compatible colors. The calculation of the C vector might be an issue.

We could have added the notions of distance and light constraints mentioned in the paper of S. Bazeille and I. Quidu and L. Jaulin [1] to add robustness to the system.

With more time and means, it would have been desirable to capture more precise images, for example at measured distances and with a more stable BlueROV (the video was captured at a time in our Guerledan project when our robot was oscillating a lot, because we were writing our own stabilization).

With such captures, we could train a CNN model with several colors at several distances, which could enable us to use this concept of compatible colors on cooler colors, which currently poses difficulties. We could also make a mix of both.

References

[1] S. Bazeille and I. Quidu and L. Jaulin, "Color-based underwater object recognition using water light attenuation" , in Journal of Intelligent Service Robotics vol.5, 2012

[2] J. Ahlen, "Color correction of underwater images using spectral data," Ph.D. dissertation, Uppsala University, Centre for Image Analysis, 2005

[3] S. Bazeille and I. Quidu and L. Jaulin "Identification of underwater man-made object using a colour criterion." in Proceedings of the Institute of acoustics, vol 29,2007

[4] Practices and pitfalls in inferring neural representations - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/a-the-RGB-color-space-black-arrows-show-the-three-main-color-dimensions-whose-values_fig2_323952018 [accessed 27 Feb, 2024]

[5] H. R. Gordon, "Can the lambert-beer law be applied to the diffuse attenuation coefficient of ocean water," Limnology and Oceanography, vol. 34, 1989.

[6] Wang, K.; Hu, Y.; Chen, J.; Wu, X.; Zhao, X.; Li, Y. Underwater Image Restoration Based on a Parallel Convolutional Neural Network. Remote Sens. 2019, 11, 1591. https://doi.org/10.3390/rs11131591

Use of the Kalman filter for an efficient tracking of an underwater robot equipped with an USBL

Tristan Le Floch

February 2023

1 Abstract

The aim of this research is to improve ultra-short baseline (USBL) underwater positioning accuracy, vital in navigation. Despite its simplicity and cost-effectiveness, environmental noise affects accuracy, leading to significant positioning errors. These errors can impact decision-making in subsequent processing. The research aims first, to identify error sources and reducing noise impact on measures. There are various possibilities to enhance accuracy, two of them are : a USBL system integrating Kalman filtering and/or integrating Interval analysis. Kalman filtering enables precise determination of the underwater robot's coordinates. Simulation evaluations affirm the efficiency of this USBL positioning method, demonstrating significant enhancement in accuracy.

2 Introduction

The main objective in this research is to find a precise localization method for underwater robots.[Lou21] offers a precise insight on underwater localization and its possibilities. The interval analysis is hard to apply to USBL measurements, which are a lot subject to outliers, as the main utility of this method is to ensure the knowledge of every possible position with good accuracy. On the other hand, the Kalman filter doesn't need a strict interval to manage the uncertainties over the measurements, as this method is probabilistic. So it comes very handful, coupled with an outliers filter to process a good underwater localization. In this document, we will see how USBL sensors combined with Outliers filtering and Kalman Filtering can achieve underwater localization.

Contents

3 Kalman Filter

Let's note $\mathbf{x}$ the state of the system.

We need to modelize the system with state equations:

$$\begin{cases} \dot{\mathbf{x}} = f_c(\mathbf{x}, \mathbf{u}) \\ \mathbf{y} = g(\mathbf{x}) \end{cases}$$

with$\mathbf{u}$ the commands sent to the robot.

We can then, discretize the system, for instance with the Euler method. Therefore, we obtain the following system which also consider the noises in the prediction and observation which are denoted α_k and β_k:

$$\begin{cases} \mathbf{x}_{k+1} = \mathbf{x}_k + dt \cdot f_c(\mathbf{x}_k, \mathbf{u}_k) + \alpha_k = f(\mathbf{x}_k, \mathbf{u}_k) + \alpha_k \\ \mathbf{y}_k = g(\mathbf{x}_k) + \beta_k \end{cases}$$

In order to apply the Kalman filter, we need f and g to be linear and the noises α_k and β_k to be gaussians. Let's state $\Gamma_{\alpha_k} = dt * \Gamma_\alpha$ and Γ_{β_k} the covariances matrices associated to the noises α_k and β_k.

We can therefore, write our system under the following linear form:

$$\begin{cases} \mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) + \boldsymbol{\alpha}_k = \mathbf{A}_k \mathbf{x}_k + \mathbf{u}_k + \alpha_k \\ \mathbf{y}_k = g(\mathbf{x}_k) + \boldsymbol{\beta}_k = \mathbf{C}_k \mathbf{x}_k + \boldsymbol{\beta}_k \end{cases}$$

To end with, we can apply the Kalman Filter, we denote $\hat{\mathbf{x}}_{k|k-1}$ the estimation of $\mathbf{x}_k$ considering all the past measurements (0,...,$k-1$). Then we receive the k-th measurement which allows us to correct the estimation and obtain $\hat{\mathbf{x}}_{k|k}$. We finally run the prediction step to obtain $\hat{\mathbf{x}}_{k+1|k}$. We also consider $\Gamma_{k|k-1}$, $\Gamma_{k|k}$ and $\Gamma_{k+1|k}$ for the uncertainty on the estimation $\hat{\mathbf{x}}$ at every step of the filtering.

Algorithm 1: Kalman Filter

Data: $\hat{\mathbf{x}}_{k|k-1}, \boldsymbol{\Gamma}_{k|k-1}, \mathbf{y}_k, \mathbf{u}_k$
Result: $\hat{\mathbf{x}}_{k+1|k}, \boldsymbol{\Gamma}_{k+1|k}$

1 $\mathbf{S}_k = \mathbf{C}_k \boldsymbol{\Gamma}_{k|k-1} \mathbf{C}_k^T + \boldsymbol{\Gamma}_{\beta_k}$
2 $\mathbf{K}_k = \boldsymbol{\Gamma}_{k|k-1} \mathbf{C}_k^T \mathbf{S}_k^{-1}$
3 $\tilde{\mathbf{y}}_k = \mathbf{y}_k - \mathbf{C}_k \hat{\mathbf{x}}_{k|k-1}$
4 $\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k$
5 $\boldsymbol{\Gamma}_{k|k} = (I - \mathbf{K}_k \mathbf{C}_k) \boldsymbol{\Gamma}_{k|k-1}$
6 $\hat{\mathbf{x}}_{k+1|k} = \mathbf{A}_k \hat{\mathbf{x}}_{k|k} + \mathbf{u}_k = f(\hat{\mathbf{x}}_{k|k}, \mathbf{u}_k)$
7 $\boldsymbol{\Gamma}_{k+1|k} = \mathbf{A}_k \boldsymbol{\Gamma}_{k|k} \mathbf{A}_k^T + \boldsymbol{\Gamma}_{\alpha_k}$

Steps 1 to 5 correspond to the correction, and steps 6 to 7 correspond to the prediction.

4 Underwater localization with a Kalman Filter

4.1 Description of the system

The advantage of the Kalman Filter is its ability to correct the estimation over various measurements, it can merge data from all sensors on the robot (for instance: USBL, IMU, pressure sensor, compass, ...).

Here we consider an AUV equipped with an USBL but also with a compass which grants a high-precision knowledge about the yaw of the AUV.

4.2 Model

In this research we wish to apply the Kalman Filter to a simple system in the 2D, as our main objective is to to focus on the Kalman Filter application to the USBL. So, we chose to modelize our system as a Dubins car:

$$\mathbf{x} = \begin{pmatrix} p_x \\ p_y \\ \psi \\ v \end{pmatrix}$$

with p_x and p_y the coordinates of the robot, v its speed and ψ its yaw.
And we have the following evolution function:

$$\dot{\mathbf{x}} = f_c(\mathbf{x}, \mathbf{u}) = \begin{pmatrix} v * cos(\psi) \\ v * sin(\psi) \\ u1 \\ u2 \end{pmatrix}$$

Therefore, the discrete state equations are:

$$\begin{cases} \mathbf{x}_{k+1} = \mathbf{x}_k + dt \cdot f_c(\mathbf{x}_k, \mathbf{u}_k) + \alpha_k = f(\mathbf{x}_k, \mathbf{u}_k) + \alpha_k \\ \mathbf{y}_k = g(\mathbf{x}_k) + \beta_k \end{cases}$$

4.3 Observation function

For the reference USBL we have the state:

$$\mathbf{x}_{ref} = \begin{pmatrix} x_{ref} \\ y_{ref} \\ \psi_{ref} \end{pmatrix}$$

The USBLs allow to measure the distance d to the robot and the relative bearing δ between the direction of the reference USBLS and the one of the

embedded USBL, so we deduce:

$$\begin{pmatrix} p_x \\ p_y \end{pmatrix} = \begin{pmatrix} x_{ref} \\ y_{ref} \end{pmatrix} + d \begin{pmatrix} cos(\delta + \psi_{ref}) \\ sin(\delta + \psi_{ref}) \end{pmatrix}$$

We add the measurement of the yaw, for the Kalman filter to merge the data, and we have the following observation function:

$$\mathbf{y} = \begin{pmatrix} p_x \\ p_y \\ \psi \end{pmatrix} = g(\mathbf{x})$$

The observation function g is linear, so we can express the observation function in a linear form:

$$\begin{cases} \mathbf{y}_k = \mathbf{C}_k \mathbf{x}_k + \boldsymbol{\beta}_k \\ \mathbf{C}_k = \mathbf{C} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \end{cases}$$

4.4 Extended Kalman Filter

The evolution function is non-linear so we need to use an Extended Kalman Filter by linearizing the evolution function. We process as following:

We assume that we have a state estimation $\hat{\mathbf{x}}_k$ of $\mathbf{x}_k$, we can therefore linearize the evolution function around this vector:

$$\mathbf{x}_{k+1} = f(\hat{\mathbf{x}}_k, \mathbf{u}_k) + \frac{\partial f(\hat{\mathbf{x}}_k, \mathbf{u}_k)}{\partial x} \cdot (\mathbf{x}_k - \hat{\mathbf{x}}_k) + \boldsymbol{\alpha}_k$$

From this we deduce the following linearized system:

$$\mathbf{x}_{k+1} = \mathbf{A}_k \mathbf{x}_k + \mathbf{v}_k + \boldsymbol{\alpha}_k$$

with:

$$\begin{cases} \mathbf{A}_k = \frac{\partial f(\hat{\mathbf{x}}_k, \mathbf{u}_k)}{\partial x} = \begin{pmatrix} 1 & 0 & -dt * \hat{v}_k * sin(\hat{\psi}_k) & dt * cos(\hat{\psi}_k) \\ 0 & 1 & dt * \hat{v}_k * cos(\hat{\psi}_k) & dt * sin(\hat{\psi}_k) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\ \mathbf{v}_k = f(\hat{\mathbf{x}}_k, \mathbf{u}_k) - \mathbf{A}_k \hat{\mathbf{x}}_k \end{cases}$$

Then we can apply the Extended Kalman Filter. We know $\hat{\mathbf{x}}_{k|k-1}$ the estimate of $\mathbf{x}_k$ considering all the past measurements. OWe receive the k-th measurement which allows us to process the correction step and obtain $\hat{\mathbf{x}}_{k|k}$.

The next step is to linearize our system around the best estimation we have of the state at the current time: $\hat{\mathbf{x}}_{k|k}$. Finally we process the prediction step with the Extended Kalman Filter and we obtain $\hat{\mathbf{x}}_{k+1|k}$.

Algorithm 2: Extended Kalman Filter

Data: $\hat{\mathbf{x}}_{k|k-1}, \mathbf{\Gamma}_{k|k-1}, \mathbf{y}_k, \mathbf{u}_k$
Result: $\hat{\mathbf{x}}_{k+1|k}, \mathbf{\Gamma}_{k+1|k}$

1 $\mathbf{S}_k = \mathbf{C}_k \mathbf{\Gamma}_{k|k-1} \mathbf{C}_k^T + \mathbf{\Gamma}_{\beta_k}$
2 $\mathbf{K}_k = \mathbf{\Gamma}_{k|k-1} \mathbf{C}_k^T \mathbf{S}_k^{-1}$
3 $\tilde{\mathbf{y}}_k = \mathbf{y}_k - \mathbf{C}_k \hat{\mathbf{x}}_{k|k-1}$
4 $\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k$
5 $\mathbf{\Gamma}_{k|k} = (I - \mathbf{K}_k \mathbf{C}_k) \mathbf{\Gamma}_{k|k-1}$

6 $\mathbf{A}_k = \frac{\partial f(\hat{\mathbf{x}}_{k|k}, \mathbf{u}_k)}{\partial x}$

7 $\hat{\mathbf{x}}_{k+1|k} = \mathbf{A}_k \hat{\mathbf{x}}_{k|k} + \mathbf{v}_k = f(\hat{\mathbf{x}}_{k|k}, \mathbf{u}_k)$
8 $\mathbf{\Gamma}_{k+1|k} = \mathbf{A}_k \mathbf{\Gamma}_{k|k} \mathbf{A}_k^T + \mathbf{\Gamma}_{\alpha_k}$

Steps 1 to 5 correspond to the correction, step 6 corresponds to the linearization and steps 7 to 8 correspond to the prediction.

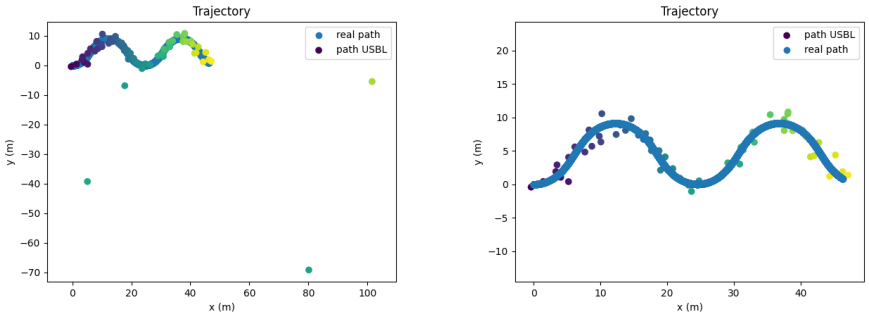
5 Outliers filtering

The main disadvantage of USBLs is that they are not only subject to their intrinsic noises but also to perturbations from the environment. For instance, the USBL can receive echos from previous pings and consider it is a new one, or a ping can also not reach the second USBL due to obstacles or reflections. Indeed, multiple acoustic phenomenon can induce biased measurements, which are far further from the reality than the intrinsic uncertainties indicate it could be. So if they are not considered as biased in the Kalman filter, it will process them as a right measurement with its uncertainties, and it will falsify the estimation. These wrong measurements are called outliers. They are illustrated and dealt with in [Luo+20].

Before applying the Kalman filter to a new measurement, we have to apply an outliers filter in order to make sure it will not wrong the estimation.

For this purpose we computed a simple outlier filter which is based on the initial position of the robot and its speed. When we receive a new measurement, we simply look at the previous position and the speed of the robot. If the measured position is not coherent with previous one considering the speed of the robot, it is an outlier and we filter it.

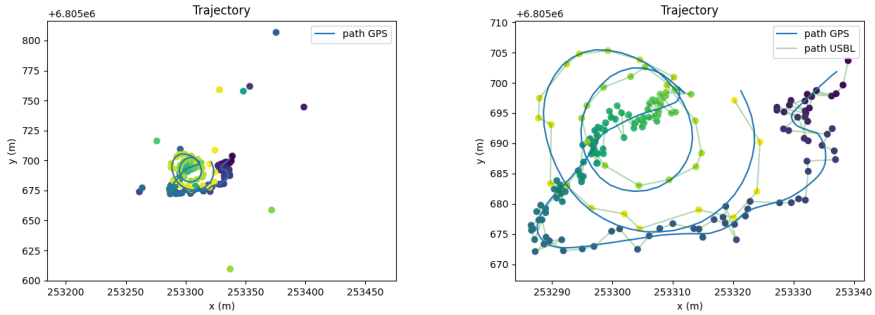
Let's observe its effect on a simulated simple trajectory and a real one :



(a) Measurements with outliers

(b) Filtered measurements

Figure 1: Simulated trajectory



(a) Measurements with outliers

(b) Filtered measurements

Figure 2: Real trajectory

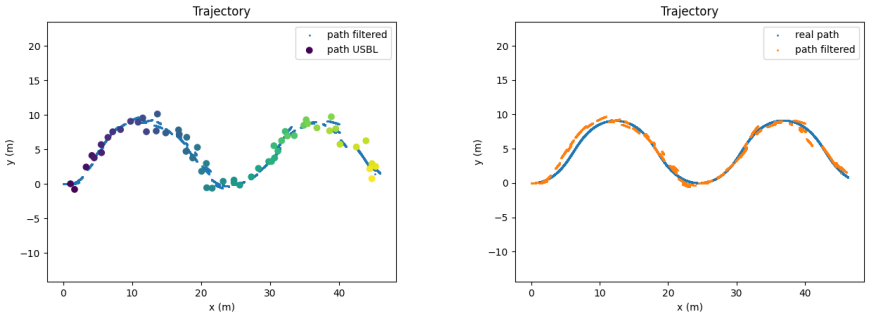
6 Simulation

6.1 Real-time filtering

Simulation of Kalman filtering for the localization of a robot in the plan with USBL and compass measurement. Simulation Kalman

Here we add a higher noise on USBL and yaw measurements than it is in reality, this pessimism allow us to prove the efficiency of the Kalman filter and to better illustrates its processing. But in reality we would know precisely the yaw.

6.2 Post-processing



(a) with measurements

(b) with real trajectory

Figure 3: Filtered trajectory

References

[Luo+20] Qinghua Luo et al. “An Ultra-Short Baseline Underwater Positioning System with Kalman Filtering”. In: (2020). URL: https://www.mdpi.com/1424-8220/21/1/143.

[Lou21] Luc Jaulin Louédec Morgan. “Interval Extended Kalman Filter—Application to Underwater Localization and Control”. In: (2021). URL: https://www.ensta-bretagne.fr/jaulin/paper_iekf.pdf.

Visual SLAM : implentation of a monocular method

J. Le Gouallec*
*Ensta Bretagne, 2 rue François Verny, 29200 Brest, France

***Abstract*—Autonomous robots are increasingly used for rescue missions, inspecting historic sites, monitoring industrial sites or wildlife. However, most of these autopilot missions require GNSS data to operate, or the fusion of multiple sensors, which adds to the robot's weight and reduces its autonomy. However, in rescue applications or in unstructured environments such as tunnels or forests, maps and GPS quality may be lacking hence the need for robust visual positioning methods. In my study, I propose to compare 2 visual SLAM techniques : Monocular Slam and RGB-D Slam. The aim is to see whether these techniques deliver sufficiently high accuracy to enable navigation in unknown environments in the absence of GNSS data and without clear markers, such as roads or streets. For testing purposes, I'll be using the open source database https://sites.google.com/view/awesome-slam-datasets/.**

***Index Terms*—Visual Slam, Monocular Slam, RGB-D Slam, Deadreckoning.**

I. Introduction

SLAM is a fundamental important problem in mobile robotics. It is a technique for obtaining a 3D reconstruction of an unknown environment. SLAM stands for Simultaneous Localisation And Mapping, which means that the robot both constructs a map of the scene and positions itself in this map. The problem is paradoxical because a map is needed to define location and location is needed to build a map. The visual SLAM is a technique based on visual information only. The idea behind this approach is that observing the same point of interest from different viewpoints enables us to estimate its 3D position. It is thus possible to reconstruct a map of the environment over time in the form of a 3D point cloud. Since the 2000s, many works have been developed to tackle this problem and it is still today a very active field of research.

In 2006, [1] proposed a real-time method based only on the video stream captured by a single camera, whose intrinsic parameters (focal distance, optic center, radial distortion) are known. It relies on identifying key points from different views, and computing the relatives poses of the cameras as well as the 3D coordinates of the interest points. An optimisation of the pose computation is performed through Levenberg-Marquardt algorithm. This approach enables a robust estimation by curbing the propagation of uncertainties between poses. The method has shown an accuracy from 2 meter to below one meter on missions, with a vehicle whose velocity is about 1m/s.

In 2014, [2] implemented a technique that takes into account the geometry of the scene in order to detect the points that fit the model. In order to overcome the limitations of monocular SLAM, it integrate additional information to the classical approaches. The geometric used are often 2D or 3D models of the buildings in the buildings in the area explored. Indeed, in urban environments, buildings represent the most widely observed geometric structures. With facades oriented in different directions orthogonal to the road planes, they provide the geometric information needed to constrain the degrees of freedom in the SLAM reconstruction plane.

In 2017, [3] established a review of RGB-D SLAM in real-time. Unlike traditional SLAM methods that use only visual information (RGB), RGB-D SLAM incorporates depth information (D), typically obtained from depth sensors like Microsoft Kinect or LiDAR scanners, in addition to RGB data. With a single camera and no a priori knowledge of the scene geometry, visual SLAM makes it possible to locate the camera to within one scale factor. The scale factor is fixed initially, but drifts over time. Monocular visual SLAM requires at least three key images to initialize using using the 5-point algorithm proposed by Nister. The availability of a depth map makes this step unnecessary. Only one image and one depth map are needed to build the initial point cloud. One main challenge is to improve the localisation of the robot, without deteriorating its computational performance. Nevertheless RGBD-SLAM remains sensitive to sudden movements, large rotations and lack of texture.

In 2020, [4] presented an real-time approach that exploits semantic information present in the environment. The semantic information refers to the different objects that are identified in the scene such as a chair, a table, a bin (etc.). This makes it possible to create distinctive landmarks. The method mixes low-level visual odometry and geometrical information corresponding to planar surfaces extracted from detected semantic objects. The semantic objects can be detected using state of the art object based detectors such as YoloV3 which satisfies the on board computational limitations of most mobile robots.

The same year, [5] proposed a method based on reinforcement learning. It provides an overview of visual navigation for artificial agents, particularly focusing on its integration with deep reinforcement learning (DRL). It systematically discusses various categories of visual DRL navigation algorithms, including direct, hierarchical, multi-task, memory-inference, and vision-language approaches. The objective of reinforcement learning is to maximize the total reward obtained over time, which is calculated as a discounted sum of individual rewards. Through interactions with the environment, RL agents aim to learn the relationship between the state of the environment and the actions they take.

In the next parts, we will keep our attention on monocular SLAM in order to try an implementation and assess the performance of the model.

II. Description of the algorithm

In this section, we will focus on of monocular SLAM and RGB-D SLAM and will dive deeper into their algorithm in order to implement them in the next part.

A. Monocular SLAM

Points of interests are located in the frame at instant t and t+1 using Harris corners detection [6]. It relies on analyzing variations in intensity in small local neighborhoods of pixels. It calculates the intensity variations in all directions for each pixel and computes a score based on the amount of change in intensity that occurs when a small shift is applied to a window around the pixel. High scores indicate corners or key points. Then, SIMD matching algorithm compares the features of the key points to identify their displacement from one frame to another. To prevent performing the algorithm on all the frames, images relatively far from each other are selected (with a minimum of matching points). The coordinate system associated to the first frame is taken as the reference of the 3D map. The relative poses of the others frames are computed relying on epipolar geometry.
The Essential matrix E verifies :

$$p_{frame1} \cdot E \cdot p_{frame2} = 0 \quad (1)$$

$$\Leftrightarrow \quad B \cdot E = 0 \quad (2)$$

The equation (2) is resolved using RANSAC (Random Sample Consensus) which is a robust method that gives correct results despite the presence of outliers in pairs of interest points.
From E, it is possible to recover the relative rotation and translation by decomposition (3):

$$E = A \cdot R \quad (3)$$

Where R is the orientation matrix of the camera and T the translation vector :

$$A = \begin{pmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{pmatrix} \quad (4)$$

Once theses matrix are computed, we obtain the extrinsic camera matrix of the t+1 pose (5):

$$M = \begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix} \quad (5)$$

From this matrix and the intrinsic camera matrix (6) with f as the focal distance and c as the optic center :

$$K = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \quad (6)$$

The homogeneous 2D coordinates (u,v,w) and the 3D coordinates (x,y,z) of the key points verify the equation (7) :

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = K \cdot M \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \quad (7)$$

From (7), it is possible to establish a Direct Linear Transform (DLT) system, which is resolved using SVD method.

Since we have obtained the relatives poses of the camera matrix and the 3D coordinates of the key points, it would be possible to estimate the successive poses of the camera during the mission. But we prefer not doing it incrementally between views t, t+1, t+2 etc. because, reprojection errors will only increase.

That is why [1] uses the bundle adjustment method which uses many views information to reduce the error on pose estimations and triangulation of key points. It optimizes camera poses and 3D structure by minimizing reprojection errors. The process involves iteratively refining initial estimates of camera poses and 3D points. At each iteration, the algorithm computes gradients of the reprojection error with respect to camera poses and 3D points. These gradients guide the optimization process, which typically employs non-linear optimization methods like Gauss-Newton or Levenberg-Marquardt. Bundle adjustment simultaneously adjusts all camera poses and 3D points, ensuring global consistency. This iterative refinement continues until convergence, resulting in more accurate reconstructions of the environment and more precise camera trajectory estimations.

III. Experiments and results

In order to assess the performance of the model, I tried to implement in python the algorithm presented in part II on the dataset ”rgbd_dataset_fr2_pioneer_slam”. Many functions are already implemented in OpenCV library, which makes the job a little bit easier. I relied on course exercises and open source tutorials.

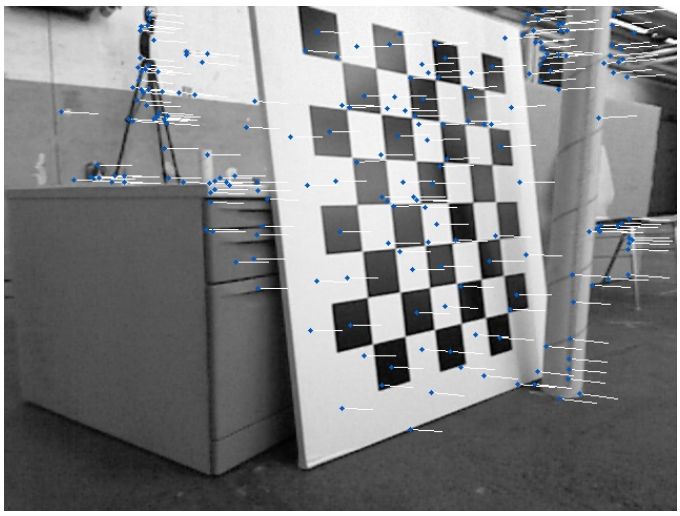


Fig. 1. Optic flow detected with SIFT algorithm between frame 1 and 25.

Unlike [1], I computed the interest points and their features with SIFT (Scale-Invariant Feature Transform), which is a more robust than Harris corners' because it is less affected by scale, rotation, or luminosity variations. A Brute Force matcher is used to compare the features and make pairs of points. The Fig. 1 show the displacement of the interest points tracked between the considered frames. The blue point represents the key point in the second frame, and the white line is the travel from the first frame. Only inlier points are represented following to the computation of the Essential matrix with RANSAC.

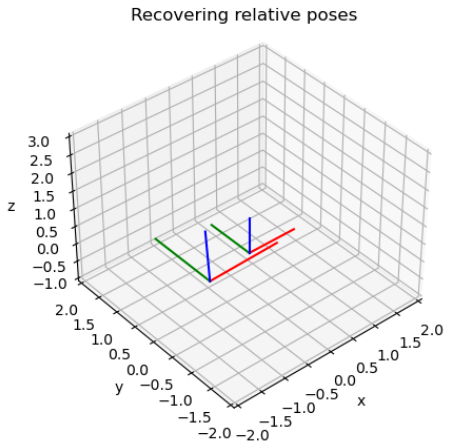


Fig. 2. Relative poses computed from the key points

From the Essential matrix, I computed the relative poses of the cameras. On Fig. 2, we witness that the model understands the displacement as a translation and not as a rotation as I observed watching the video flow. Indeed, the robot seems to be rotation along its z axis between theses 2 frames but as mentioned in the Vision3D class this movement can be easily mistaken.

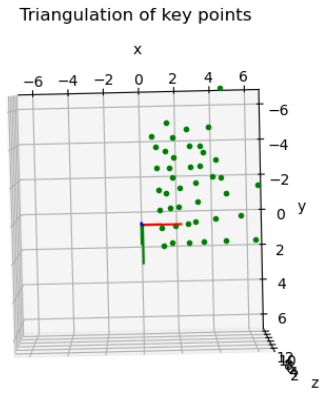


Fig. 3. Triangulation of the key points.

I tried different approaches so as to identify the problem. I recomputed the triangulation based on a stéreo vision approach with this tutorial: "https://temugeb.github.io/opencv/python/2021/02/02/stereo-camera-calibration-and-triangulation.html". The result is shown on Fig. 3. This time, the cameras seems to have a slight relative rotation so it is a more appropriate result. Therefore, I tried to triangulate points solving the DLT system previously mentioned in part II, using SVD method. The 3D key points are displayed on Fig. 3. to within one scale factor. I was stuck at this part because I couldn't obtain an acceptable triangulation. If I had obtained a first decent estimation of the trajectory, I could have plotted it on Fig. 4. I didn't try to implement the Levenberg-Marquardt algorithm either because the results of the elementary part of the method I implemented were absurd.

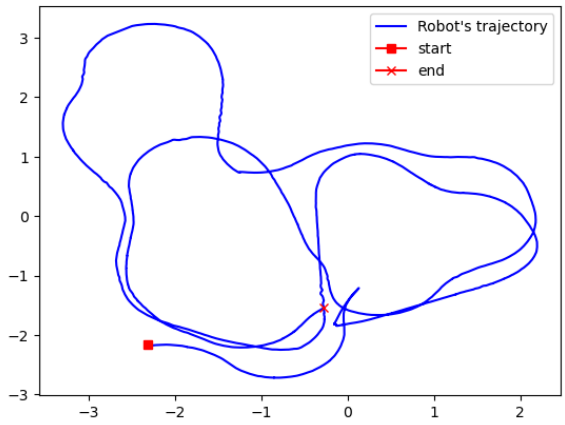


Fig. 4. Reference trajectory of the robot

IV. Conclusion

In this paper, I presented several methods of visual SLAMs. I tried to implement the most basic approach but I encountered a lot of difficulties. I am a bit disappointed of my results because I got stuck on a fundamental part which prevented me from implementing the optimisation algorithm.

References

1 Mouragnon, E., Lhuillier, M., Dhome, M., Dekeyser, F., and Sayd, P., "Real time localization and 3d reconstruction." Université Blaise Pascal/CNRS - CEA/LIST/DTSI/SARC, 2006.

2 Larnaout, D., "Localisation d'un véhicule à l'aide d'un slam visuel contraint." Université Blaise Pascal - Clermont-Ferrand, 2014.

3 Melbouci, K., "Contributions au rgbd-slam." Université Clermont Auvergne, 2017.

4 BAVLE, H., DE LA PUENTE, P., P.HOW, J., and CAMPOY, P., "Vps-slam: Visual planar semantic slam for aerial robotic systems." IEEE, 2020.

5 ZENG, F., WANG, C., and SAM GE, Z., "A survey on visual navigation for artificial agents with deep reinforcement learning." IEEE, 2020.

6 Harris, C. and Stephens, M. a., "A combined corner and edge detector," in *Alvey Vision Conference*, 1998, pp. 147–151.

Rank prediction at ensta bretagne using personality traits of students

Leroy Hippolyte
ROB24
Ensta Bretagne
France
hippolyte.leroy@ensta-bretagne.org

***Abstract*—This study deals with the links between the grades of a sample of students and their personality traits. The data exploitation should have been carried out using interval calculations, but for various reasons this was not possible. The interval calculation strategy is nevertheless described in this article. the work accomplished in this study is as follows: The students of the rob24 class were asked to answer a survey about their own personality traits based on different criteria. Based on their answers and different techniques, we can rank them and then check whether this model was relevant for each student. The results are mixed: for many students, the rank predictions for each UE (teaching unit) were close to reality, but for some, much less so. This can be explained in a number of ways, including the relevance of the criteria, the calculation of scores, the difficulty of discriminating between students, the randomness of understanding the survey questions, and the small sample size, as not all the class took the survey. Note that real rankings for the units that appear on the school reports are calculated by Ensta Bretagne using the grades of the entire promotion.**

Based on their answers, and different techniques, soft skills or hard skills scores are calculated, and penalties may be applied. Then for each UE of the 2midyear, a new scored is calculated based on previous scores with different weights and sorted among the whole class.

***Index Terms*—personality traits, softskills, hardskills, grades, rank, interval, machine learning, academic success**

I. Introduction

In recent decades, increasing attention has been paid to the factors influencing academic success of students, from preschool to university. This study focus on understanding the determinants of academic performance results from the recognition of the crucial importance of education in contemporary society. Indeed, education is not only a means of access to better economic and social opportunities, but is also seen as a fundamental pillar of individual and collective development.

Education is often seen as the great equalizer, offering everyone the opportunity to realize their full potential. However, there is growing evidence that students do not begin their educational journey with equal opportunities for success. Even before they enter the school gates, profound inequalities are often already present, shaped by a multitude of social, economic and individual factors.

With this in mind, understanding the determinants of academic success becomes crucial to developing more inclusive and effective educational policies. Among the many factors that are more likely to influence academic success, students personal characteristics have been of particular interest. Indeed, an individual's personality traits have been identified as key elements likely to shape their educational experiences and, by extension, their academic success.

The idea that personality traits could play a role in academic success is closely linked to the increasing recognition of the importance of individual differences in education. While education systems have traditionally focused on imparting academic knowledge and skills, it has become increasingly clear that students' socio-emotional and personal aspects also play a crucial role in their development and academic success. Of course, this study also takes into account "traditional" abilities such as **perseverance, memory, natural abilities (IQ)** etc, but it balances them with personality traits that researchers are increasingly interested in, such as the **ability to say no, self-confidence, extroversion, neuroticism**, which expresses the ability to feel negative emotions such as anxiety, depression etc.

Thus, this emerging trend to explore the link between students' personality traits and their academic success is part of a broader movement to adopt a holistic approach to education. Rather than focusing solely on raw academic achievement, researchers and educational practitioners are increasingly recognizing the importance of taking into account the psychological and personal factors that influence students educational trajectories.

From this perspective, this literature review takes a look at recent research on the link between individuals' personality traits and their success at school. By analyzing the results of published empirical studies, we will seek to better understand how students' personal characteristics can influence their academic performance and, consequently, inform educational practices aimed at promoting success for all learners.

This paper is structured as follows. II explains how the rank is predicted considering the personality traits. III outlines the personality traits studied. The results of this study are shown in IV and analyzed in V. VI is the conclusion of this paper and VII exposes how we could have gone further in this study, notably through interval computing and/or machine learning.

II. Prediction of scores based on personality traits

Concretely, in order to make the link between grades and students personality/environment, I developed the following method to be able to discriminate them according to their personality. I assign them scores based on their answers, which define whether they are more or less likely to succeed in a given area.

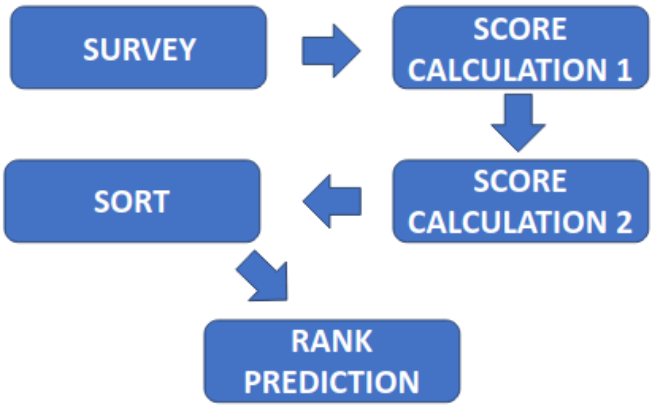


Fig. 1. Rank prediction process

So, the first step is to collect the data from the survey I've proposed, which was carried out in the following way: I ask students to rate themselves on a scale from 1 to 10 on 15 criteria I've selected. They don't know it, but these criteria fall into three categories. The first two categories are the soft and hard-skils I referred to in the introduction. There are 6 and 6 of them. Then there are penalties, of which there are 3. Then the second step is to calculate the first two hard and soft skill scores per student. We simply add the scores they've set for themselves by category and subtract certain malus depending on the soft or hard skills category that this might affect by malus category.

To give an example, I'll add the perseverance and memory scores, then subtract the frequency of partying (and drinking alcohol) score, etc. Each category is detailed in the following section.

Then, once I have these two final scores for soft and hard skills, I apply linear combinations of these scores for each UE, with different weights depending on what I consider more or less useful for each UE. Finally, I put the scores of each student per unit in lists that I sort. All that remains is to calculate the rank of each student for each UE knowing that :

Rank	Percentile
A	$\leq 15\%$
B	$16\% - 35\%$
C	$36\% - 65\%$
D	$66\% - 85\%$
E	$> 85\%$

III. Selected characteristics

As a reminder, I call hard skills the characteristics of individuals that seem obvious for academic success.

A. hard-skills

Returning to the features that give the soft and hard skills scores, I first chose the hard-skills.

- Punctuality ([6])
- Memory ([7])
- Attention to detail ([8])
- Flexibility/adaptation ([9])
- Self-discipline/perseverance ([5])
- Aptitudes (IQ/academic ability) ([5])

According to Duckworth and Seligman (2005) [5], self-discipline, like IQ, is an important predictor of students academic success. Their study reports that, in fact, self-discipline is far more important than IQ for success. As a result, I apply a higher weighting to self-discipline than to academic ability when calculating my hard-skill score.

B. soft-skills

Soft skills, by their very nature, cover personality traits and behavioral competencies that are more subjective and less tangible than hard skills, which are generally technical and measurable competencies. To select the most relevant soft skills for my study, several factors had to be taken into account. Firstly, it was essential to select soft skills that were both universal and applicable in a variety of contexts.

Moreover, the diversity of soft skills requires judicious selection to avoid redundancy and ensure coverage of the relevant dimensions of human behavior.

The criteria I have therefore chosen from the literature which seemed to be persistent are as follows :

- Ability to say no ([10])
- Sociability ([4])
- Curiosity/Conscientiousness ([1])
- Extroversion ([2])
- Self-confidence ([11])
- Stoicism ([12])

Looking at this list, we can quickly say that these criteria could indeed have an impact on student success, but what about the ability to say no? Does this characterize the fact of refusing certain things in order to stick to our own opinions, in this case we could simply link it to self-confidence, which would be redundant. In fact, in the marshmallow test [10], children are given the choice between having a short-term reward and having a better long-term one. It was shown that those who chose the better long-term reward, and therefore to say no at the present moment, had a greater chance of success than the others.

There is one factor here that is a little redundant: sociability and extroversion share the same trait, but there is a notable difference: extroverted individuals tend to be more sociable

than others, while sociable individuals aren't necessarily extroverted. In fact, I can be introverted and still be looking for lots of social interaction. That's why I've chosen to keep these two criteria, but to apply smaller coefficients to them than to the others soft-skills, because there is a certain redundancy.

C. Penalties

As I was saying earlier, 3 of the categories were actually Penalties that were applied in different ways

- Party frequencies([10])
- Financial difficulties ([13])
- Neuroticism ([3])

In fact, people who partied a lot received a penalty on their hardskills scores, and it's easy to see that this directly affects our ability to concentrate, to get up early and therefore to be punctual, and so on. As for financial difficulties, a penalty was applied to both scores [13], and finally, a penalty was applied to the hardskill scores of people with high neuroticism, since being more sensitive to stress, anxiety, depression or compassion affects the same hardskills [3].

IV. Results

Around twenty students from the ROB24 class completed the survey. Of these twenty, I had access to the rankings of about ten, which enabled me to verify the predictions. For most of them, around 70%, the results were very close to their true ranks. In almost every case, out of the 8 UE, there was 6 good predictions and 2 predictions with a difference of 1 rank. For those two predictions with 1 rank apart, both were in the same direction, either too high or too low, but never differently. For the remaining 30% of students, all predictions were wrong. Note that the predictions were much lower than their true ranks.

V. Analysis

After going back to the interval calculation, I realized that it was difficult to discriminate between the students in the robotics class, partly because we have quite similar profiles with a few differences, and I think that is why we ended up here. So it is on these few differences that we have classified the students. So the fact that we had around 70% success rate on the predictions is very promising and proves that the criteria chosen had a significant impact in determining whether a student, depending on his/her profile, was more or less likely to succeed. However, these results must be balanced against the very random nature of the small sample. In other words, how would you rate yourself on each criterion, whether it related to all your studies at ENSTA Bretagne, the present or since the start of your schooling? There was no indication other than to score on the 15 different criteria.

This explains some of the 30% very false predictions, the ones very far from the truth. In fact, for some people, a closer look at the scores they gave themselves showed that their lack of self-confidence or over-confidence affected all the answers, and that they gave themselves too much or too little in certain categories. It would have been relevant to compare the results with surveys in which students rate criteria for other students. But then, who knows us better than we do ourselves?

VI. Conclusion

In conclusion, this study aimed to investigate the relationship between students personality traits and their academic performance. Through the analysis of survey data collected from a sample of students in the ROB24 class, we sought to determine whether certain personality characteristics could predict academic success.The results of the study revealed a mixed picture. While predictions based on personality traits were fairly accurate for the majority of students, with around 70% of predictions closely matching their actual ranks in different teaching units, there were significant discrepancies for a minority of students. These discrepancies could be attributed to various factors, including the relevance of the criteria used, calculation methods, individual differences, and the small sample size. Furthermore, the inclusion of penalties for factors such as party frequency, financial difficulties, and neuroticism underscores the importance of considering both positive and negative influences on student performance. Further research is needed to redefine the methodology used in this study and explore additional factors that may influence academic success. Additionally, efforts should be made to replicate these findings in larger and more diverse samples to ensure generalizability. By gaining a deeper understanding of the complex interplay between personality traits and academic performance, educators and policymakers can develop more effective strategies to support student success and foster a more inclusive learning environment for all learners.

VII. Going further

A. Interval calculus

The initial idea of this study, which came to nothing, was to calculate students grades with intervals calculus [**?**] according to personality traits.

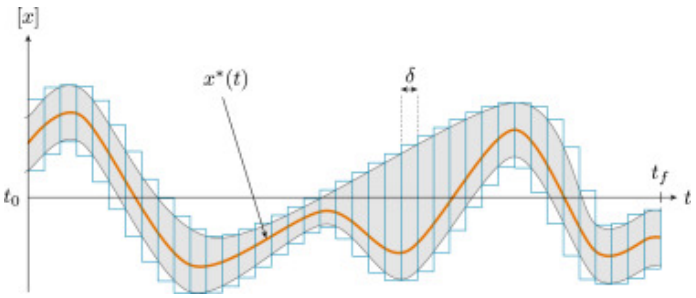


Fig. 2. tube example

The aim of interval calculation is to have a tube as reliable as possible in which we are sure we are. Here, if each slice represented a grade, it would be enough to contract for each slice on an interval where we are sure the grade is. However, how can we define a function for each discipline that converts the 15 survey data into a interval of grades? this would be too

laborious and too subjective. Now we can also reason in the same way about the UE rankings as it was done on this paper, repeat the same calculation and say that, given the assumptions made, we are more or less one rank away from the predicted rank. This is still not good, since, on the one hand, we have very false predictions as seen above, and the interval would not even include the true result, and on the other hand, since an interval of 3 ranks out of a possible one is not very reliable.

B. Machine learning

One approach that could be worth testing is machine learning. The idea is as follows, on a large graduating class, say the 1st year at ensta bretagne. We ask them about their personality traits based on the same criteria, and then, using their grade report sheets for the year, we estimate with machine learning which parameters were relevant in making them better in a given subject. This will make it possible to predict their grades in advance by asking the next promotion to answer the same survey.

REFERENCES

[1] Poropat, A. E. (2009). Conscientiousness and academic achievement: A meta-analysis. *Psychological Bulletin*, 135(1), 139.

[2] Allen, M. S., & Potkay, C. R. (2011). Extraversion and academic performance: A review and meta-analysis. *Personality and Individual Differences*, 51(7), 801-811.

[3] Richardson, M., & Abraham, C. (2009). Neuroticism and academic performance: Evidence and implications. *Personality and Individual Differences*, 47(5), 548-552.

[4] Graziano, P. A., & Tobin, R. M. (2017). Agreeableness and academic success: A systematic review. *Personality and Individual Differences*, 115, 71-83.

[5] Duckworth, A. L., & Seligman, M. E. P. (2005). Self-discipline outdoes IQ in predicting academic performance of adolescents. *Psychological Science*, 16(12), 939-944.

[6] Smith, J., & Johnson, L. (2018). The Relationship Between Punctuality and Academic Performance: A Systematic Review. *Journal of Educational Psychology*, 110(3), 354-367.

[7] Johnson, D., Smith, A., & Williams, B. (2015). The role of memory in learning. *Journal of Educational Research*, 120(2), 245-259.

[8] Brown, L., & Smith, C. (2017). Attention to detail and academic performance. *Journal of Educational Psychology*, 125(4), 512-525.

[9] Taylor, R., Johnson, M., & Davis, L. (2019). Flexibility and adaptation in educational settings. *Journal of Educational Research*, 130(1), 78-91.

[10] Mischel, W., Ebbesen, E. B., & Zeiss, A. R. (1972). Cognitive and attentional mechanisms in delay of gratification. *Journal of Personality and Social Psychology*, 21(2), 204-218.

[11] Smith, J., & Johnson, L. (2020). The Role of Self-confidence in Academic Achievement: A Longitudinal Study. *Journal of Educational Psychology*, 130(4), 532-545.

[12] Brown, A., & Taylor, R. (2021). Stoicism and Academic Success: A Cross-sectional Study. *Personality and Individual Differences*, 150, 109874.

[13] Smith, A., Johnson, L., & Brown, J. (2018). The Impact of Financial Hardship on Academic Performance: A Longitudinal Study. *Journal of Educational Psychology*, 125(3), 301-315.

[14] Simon Rohou, Luc Jaulin, Lyudmila Mihaylova, Fabrice Le Bars, Sandor M. Veres. "Guaranteed computation of robot trajectories", *Robotics and Autonomous Systems*, vol. 93, pp. 76–84, 2017.

Intruder detection and capture with a robot swarm

Théo Massa

***Abstract*—This paper tackles the intruder detection issue thanks to an automated group of robots in a close space. After introducing the context, one will try to localize a target using interval analysis. For this, the model used is described and the conception of appropriates control laws are defined in order to succeed the objective. First the target doesn't move and all the processes are done afterward, then a live algorithm is used in another coherent approach.**

I. Introduction

Allegedly to the paper that inspired this article [1], we consider n robots $R_1, ..., R_n$ at positions $a_1, ..., a_n$ and moving in a $2D$ world. In order to make the robots the more simple and cheap we can, each robot has only one sensor that detect if the intruder is in range r_{range}. This defines a visibility zone in which the robot is able to tell if there is an intruder or not. The particularity is that we only know its distance to the robot and not its relative position. Moreover, every robot is independant from the others, in the way that they don't directly communicate their position between each other. They do share some information, which will be described later, but not their position.

The group of robots, and the global system, lies in a 2D world with defined borders from which no robot can exit.

In this paper, we use a combination of set-membership approach [2] and control theory in order to compute the position of the intruder and control effectively enough the searching swarm. The set-membership approach assures that we have a zone in which we are sure to find it.

II. Model

Before going into the process used to detect and capture the intruder, it is important to describe the models used in this system. For each robot (swarm and intruder), we use a Dubin's car model:

$$\begin{cases} \dot{x} = u_1 \cos(\theta) \\ \dot{y} = u_1 \sin(\theta) \\ \dot{\theta} = u_2 \end{cases}$$

It is important to precise that here, for simplification, the intruder follows this model, but if this has to be used in the real world, we do not know for sure the precise model of the intruder, which complicates the system. However, this paper is only an introduction to this subject and deserves more work to get this application on a wider scale.

Concerning the swarm, n robots are placed randomly on the delimited zone at start and begin to move through this area in order to detect (or not) an intruder.

III. Control laws

For a robot searching for an intruder, we have to consider two situations and therefore developping two control laws. Indeed, its behaviour won't be the same if it looks for an intruder or if it has detected one and wants to follow and capture it.

A. Out of range case

Let's consider here that none of the n robots have detected the intruder. This means we are in a searching phase and we want the searching squad to cover the most area possible in order to eventually detect something. Moreover, the area is delimited and because we know that the potential intruder is in this area, we want the robots to stay in this zone. Thus, we need to find controls that will insure us that the robots will stay in the zone and cover maximum area of this zone.

In order to achieve that, the chosen strategy is to have a constant speed and turn when the robot comes close to the limits. By doing so, it may be hard to have a predefined and precise trajectory, but the robots will cover enough area to detect at each time. With those commands, we are able to obtain random but satisfying trajectory depending on the initial pose (see *Fig. 1*).

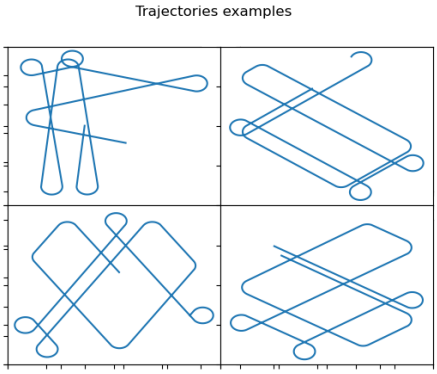


Fig. 1: Example of some trajectories

As we can see, the trajectories stays in the delimited area and, if we combine them, those trajectories covers a wide part of the surface, as seen on *Fig. 2*. We will then use those commands when it comes to find the intruder.

B. In range case

Now let's consider that at least one of the searching robots have detected that the intruder is in range. Thanks to the

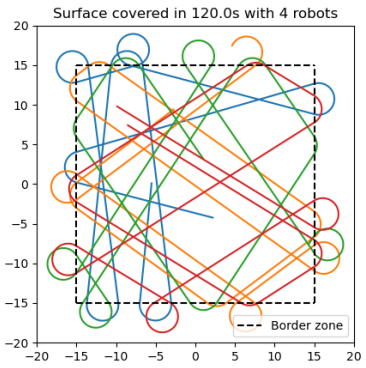


Fig. 2: Surface covered

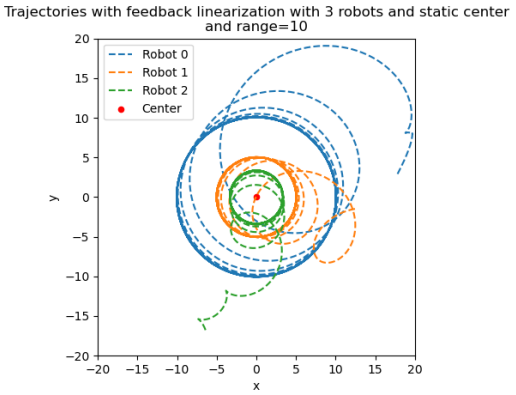


Fig. 3: Feedack linearization with static center

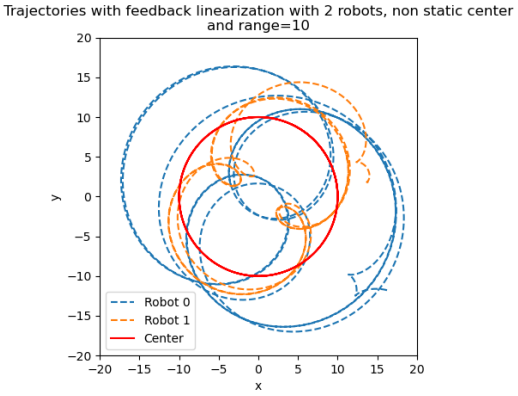


Fig. 4: Feedack linearization with non-static center

interval approach, which we will describe later, we are able to compute a zone in which we are sure that the target is. Therefore, we can have the coordinates of the center $C_z(t) = (x_z(t), y_z(t))^T$ of this zone. Those coordinates are then shared between every robot in order to use them for our control law. We want each robot to follow a circle around this point. The trajectory that the i^{th} robot needs to follow is like that:

$$\mathbf{c_i(t)} = \begin{bmatrix} x_z(t) + r_i \cos(at + \frac{2i\pi}{n}) \\ y_z(t) + r_i \sin(at + \frac{2i\pi}{n}) \\ at + \frac{2i\pi}{n} + \frac{\pi}{2} \end{bmatrix}$$

With $r_i = \frac{r_{range}}{i}$ being the radius of the trajectory around the center. This differentiation in each radius allows to improve the quality of the final detection, as it improves the intersection of the different zones obtained by each robot.

We can then use a feedback linearization method [3] [4] with $\mathbf{y} = X_i = (x_i, y_i, \theta_i)^T$ in order to get the proper $\mathbf{u} = (u_1, u_2)$. With this method we have:

$$\mathbf{u} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} (\dot{c_i}(t) + c_i(t) - y)$$

Once we have this, there is still one issue that is not solved. Indeed, obtaining precisely the derivative $\dot{c_i}$ is not direct as it depends on $\dot{x}_z(t)$ and $\dot{y}_z(t)$, which are related to the intruder behaviour, that we do not know, and the quality of the detection. To solve this issue we interpolate this derivative by considering:

$$\dot{\mathbf{x}}_z(t) = \frac{\mathbf{x}_z(t) - \mathbf{x}_z(t - k.dt)}{k.dt}$$

With $\mathbf{x}_z = (x_z, y_z)^T$ and k being the number of iterations between the actual one and the last iteration where the intruder was detected. If everything works fine, $k = 1$ at every iteration after the first detection but this is not especially true.

In *Fig 3* and *Fig 4*, we can see the efficiency of the feebdack linearisation control in both case: static and moving center.

C. Intruder law

In order to make our simulation more realistic, we need the intruder to have a coherent behaviour. We will consider that the intruder follows a potential field globally oriented toward its objective. Moreover, the intruder doesn't want to be detected by the robots. Therefore, we will consider the robots as repulsive points for the potential field of the intruder law.

Considering the two part of the potential field, the repulsive and the constant one, if M is the position of the intruder and M_i' the position of one repulsive robot, we have:

$$\overrightarrow{v} = \overrightarrow{v_c} + \sum_{i=1}^{n} \frac{\overrightarrow{M_i'M}}{\|\overrightarrow{M_i'M}\|^{\alpha+2}}$$

With α a constant that defines the repulsivity of the robot. We can then take for commands of the intruder:

$$u_{intruder} = \begin{bmatrix} v_{intruder} \\ \dot{\theta}_{intruder} \end{bmatrix} = \begin{bmatrix} \|\overrightarrow{v}\| \\ k.sawtooth(\theta_v - \theta_{intruder}) \end{bmatrix}$$

With $\theta_v = \arctan 2(v_y, v_x)$ and k being a coefficient representing the sensibility of the proportional control. The sawtooth function allows to keep the yaw difference in $[-\pi, \pi]$ (see *Fig. 5*).

IV. Intervals

Interval analysis is a method based on intervals that ensures a fiability and a security in the results [2]. This safeness is an advantage that interest us a lot for this subject.

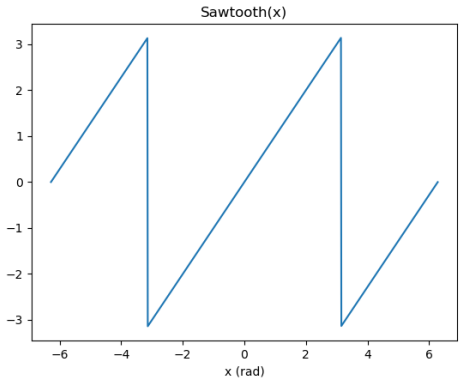


Fig. 5: Sawtooth function

A. First approach - Range-only SLAM

This part is greatly inspired from one of the tutorial of the codac library [5]. Let's consider m static intruders b_i that will be considered as landmarks, and one searching robot that moves purely in dead-reckoning. We only know its command and its initial position. The robot moves and stores the distance it measures each time the intruders are in range.

Each landmark is represented with a 2-dimension box B_i initiated to $[-\inf, \inf]^2$. Using contractions and interval properties, the objective is to reduce the most that we can these box to a surrounding of their real coordinates and, thanks to a good property of interval analysis, we will also improve the trajectory estimation.

Let's consider $\mathcal{L}^i_{dist}$ the distance constraint from a landmark b_i. With this constraint, the distance d_i is linked to the state $\mathbf{x}$:

$$\mathcal{L}^i_{dist} : d_i = \sqrt{(x_1 - b_{i,1})^2 + (x_2 - b_{i,2})^2}$$

Furthermore, $\forall k$ such that the robot detects an intruder at time t_k, let's consider the constraint:

$$\mathcal{L}^k_{eval} : p_k = x(t_k)$$

Once this is done, we have, for each time t_k and for all intruder i we have the following constraints:

$$\begin{cases} \mathcal{L}^k_{eval}(t_k, p_k, x, v) \\ \mathcal{L}^{i,k}_{dist}(p_k, B_i, d_i) \end{cases}$$

Furthermore, two constraints are also important to add, the one representing $v = f(x, u)$ and the other $v = \dot{x}$:

$$\begin{cases} \mathcal{L}_f(x, u, v) \\ \mathcal{L}_{deriv}(x, v) \end{cases}$$

Once all those constraints are added to our contraction system, we are able to localize approximately the intruders but also to improve greatly the estimated trajectory, as seen below.

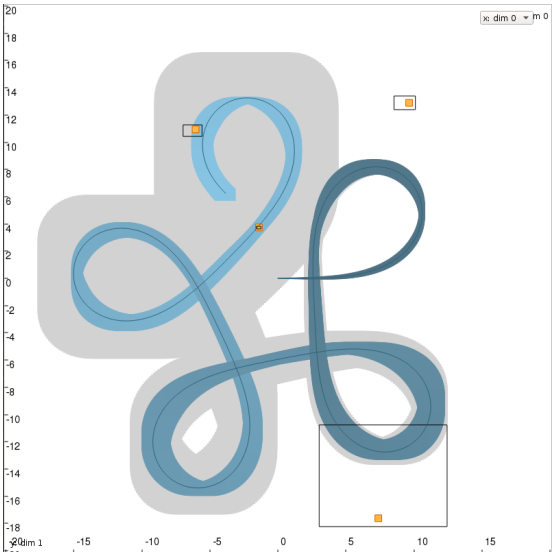


Fig. 6: Range only SLAM

B. Second approach - SIVIA Algorithm

Despite its efficiency, the main inconvenient of the first approach is that it requires post-processing to get the intruder(s) position. Even if this method allows to detect the intruder and to improve the trajectory estimation at the same time, this is hard to implement in real-time, which is what we want to do for our algorithm. Therefore, we have to find a way to be able to get the intruder position in real time.

In order to accomplish this, we have to change our point of view and do important assomptions. Starting from now, we will consider that the robot knows its state precisely everytime and that there is only one intruder. We could still do SLAM like before but live, the issue is that in order to process SLAM, the reference points, also called landmarks, needs to be statics. However, the intruder position is anything but static.

Now let's explain the full process. The intruder spawns at the right of the delimited space and moves globally toward the left. We have n robots spawned randomly on the field that begin to look for an intruder.

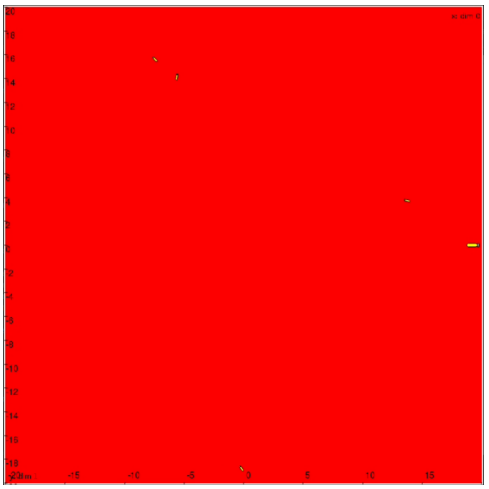


Fig. 7: Initial state

In red, this is the area in which we know the intruder is. As one can see, initially this zone covers the entirety of the zone, because we have no information on the intruder location. This is unfortunately not really clear because of the size of the robots on the image.

We are now in the search phase, which means we use the searching control. We integrate thanks to an Euler integration: $x_{k+1} = x_k + dt * f(x_k, u)$ and for each robot, we check if the intruder is in range. Once we detect the intruder, the SIVIA algorithm [6] is used to get a list of boxes from which the union of them is a zone where the intruder is sure to be found.

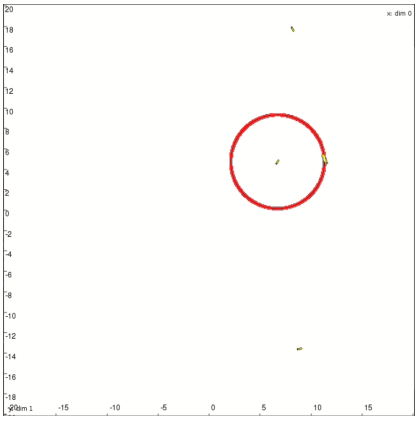


Fig. 8: First detection

This zone, if only one robot detected the intruder, is quite logically a circle, as we know only the distance between the robot and the intruder, with a certain incertitude due to the sensor imprecision. Once this zone is obtained, we can calculate its center. If the zone is composed by p boxes $B_1, ..., B_p$ of center c_i and volume v_i, we can compute the center of the global zone by computing the volume-weighted average of the centers:

$$\begin{bmatrix} x_{intruder} \\ y_{intruder} \end{bmatrix} = \frac{\sum_{i=1}^{p} v_i.c_i}{\sum_{i=1}^{p} v_i}$$

This center is then used for each robot to follow a circle around this point. The robots will then converge toward this zone, in order to augment the precision and surround the intruder. After, they will follow a round trajectory around the intruder while still following him (*Fig. 9, Fig. 10*).

V. Conclusion

The results here are quite satisfying. For each trial, the searching swarm succeeds in finding and surrounding the intruder. However, some work can still be done on this subject. As it is done by [1], we could have implemented a safe area that increases when a robot doesn't detect an intruder. By doing so, one can improve the searching strategy.

One of the main improvement is the swarm's command law. The laws that are used here are effective but there are still some issues. Indeed, depending on its initial pose, is it possible that one of the searching robot finds itself locked on the border zone. Furthermore, during the transition between searching and surrounding, the swarm reacts way too fast and their behavior is not coherent at all. They try to rejoin their circle trajectory with too much speed, which does not reflect a realistic behaviour. Moreover, for now the intruder's trajectory is just deviated by the surrounding swarm but it could be interesting to block the intruders movements. Finally, it could be interesting to implement a real swarm behaviour for the searching squad, as they are quite independant for now.

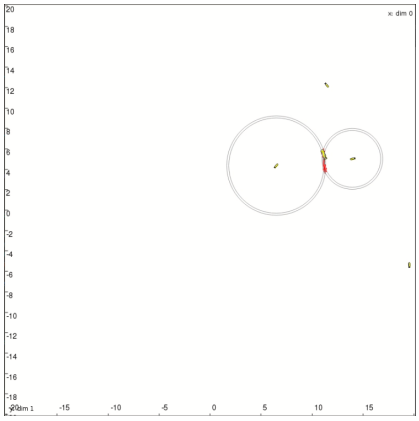


Fig. 9: Second robot arrived

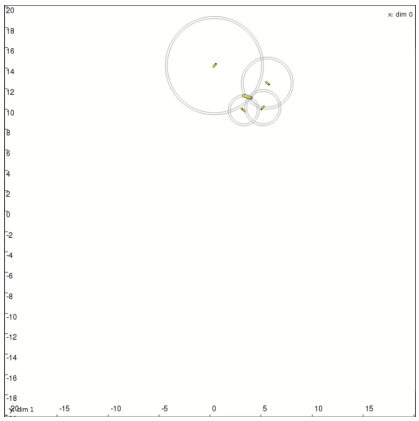


Fig. 10: Surrounding the intruder

References

[1] K. Vencatasamy, L. Jaulin, and B. Zerr, "Secure the zone from intruders with a group robots."

[2] J. P. Merlet, "Interval analysis and robotics," in *Robotics Research*, ser. Springer Tracts in Advanced Robotics, M. Kaneko and Y. Nakamura, Eds. Berlin, Heidelberg: Springer, 2011, pp. 147–156.

[3] A. J. Krener, *Feedback Linearization*. New York, NY: Springer New York, 1999, pp. 66–98. [Online]. Available: https://doi.org/10.1007/978-1-4612-1416-8_3

[4] B. d'Andréa Novel, G. Campion, and G. Bastin, "Control of nonholonomic wheeled mobile robots by state feedback linearization," *The International Journal of Robotics Research*, vol. 14, no. 6, pp. 543–559, 1995. [Online]. Available: https://doi.org/10.1177/027836499501400602

[5] S. Rohou, B. Desrochers *et al.*, "The Codac library – Constraint-programming for robotics," 2022, http://codac.io.

[6] P. Herrero, P. Georgiou, C. Toumazou, B. Delaunay, and L. Jaulin, "An Efficient Implementation of SIVIA Algorithm in a High-Level Numerical Programming Language," *Reliable Computing*, 2012.

Soft robotics and its applications : state of the art

Marguerite MIALLIER
ROB 24
ENSTA Bretagne
Brest, France
marguerite.miallier@ensta-bretagne.org

***Abstract*—Appeared at the beginning of the 21st century, soft robotics has made great progress. This article presents current solutions regarding the materials used, such as polymers, modeling and control methods, as well as the current uses of these robots, often bio-inspired.**

***Index Terms*—soft robotics, polymers, bio-inspiration, soft electronics, control**

I. Introduction

Soft robotics is a subfield of robotics, which appeared at the beginning of the 21st century and uses flexible materials such as rubber, polymers, or springs. There are numerous applications, particularly for the exploration of soft environments (living organisms for example) for surgery, or the manipulation of fragile objects. These robots often use bio-inspired materials and actuators, very different from what can be found in rigid robotics. This therefore involves developing new control methods. In this article we will propose a state of the art of soft robotics: robot design, modeling, control and applications.

II. Soft robot design

A. Materials

First, it must be clarified that what is "soft" is the body of the robot. To build this type of robot, you must therefore use soft materials. These materials are characterized by a Young's modulus close to that of soft biological materials (such as skin) [16] . The use of such materials makes it possible to reduce the risk of injuries in human-robot interaction, for example. Some elastic materials commonly used for the design of soft robots are Sylgard 184, a silicone used for electrical and electronical applications, Smooth-Sil 950 and EcoFlex 00-30 (REF) [17] which are platinum silicones used for multiple molding applications like candles or food.

Another important parameter in the choice of material is its viscoelasticity. A purely elastic material will not dissipate energy, whereas a viscous material will be able to dissipate energy and thus maintain stable movement despite the application of forces [2]. Examples of viscous materials that can be used in soft robots are polyethylene glycols (PEG) hydrogels and polydimethylsiloxane (PDMS).

B. Actuators

While rigid robots are most of the time motorized, other actuation systems must be used for soft robots, compatible with the materials used. These methods are often less precise that conventionnal actuators for rigid robots, but they allow more flexibility [2]. Variable length tendons can be embedded in short segments to create robotic arms like octopus arms [18], but also other methods such as pneumatic or electrical actuation, or even chemical stimulation can be used. More details about these methods will be given in this section.

1) Stiffness variation: A soft robot can be actuated by modifying its stiffness. This can be done by filling an elastic bladder with small beads for example. By evacuating the bladder, the beads will tighten until the system moves to a solid-like state. This method is called particle jamming [1] Robots using this method can achieve a large number of different shapes, and can support large loads thanks to their ability to solidify.

Another solution to change the stiffness of the robot is to embed thermoformable materials [2] such as metal, and heaters. That way, the stiffness can be reduced on-the-fly by heating the material.

2) Strain-mismatch: A popular way to control soft robots is to exploit the deformations induced by strain-mismatch. Some materials used for this actuation method are dielectric polymers [10], shape memory alloys, ionic polymer metal composites or polydimethylsiloxane (PDMS) [2]. The deformation can be induced in both hard materials (metal for example) in thin layers, soft materials like PDMS, and hybrid systems. This deformations can be non linear, if induced by chemical processes or diffusion, which is interesting for actuation.

3) Pneumatics: Soft robots can be actuated by inflating air in elastic materials, allowing to create deformations. Pneumatic artificial muscles (PAMs) [16] are examples of linear soft actuators composed of elastomer tubes in fiber sleeves. Fluidic elastomer actuators (FEAs) [16] are a type of highly extensible and adaptable, low-power soft actuator. FEAs are actuated by increasing the pressure in the embedded elastomer channels, which expands them.

The main problem with pneumatic networks actuation is that it can be very slow due to the high strains needed (several seconds to achieve the desired position). This problem has been studied and a solution has been developed, presented in [9]. It consists of the design of pneumatic networks with a particular structure, which allow rapid control with a lower risk of rupture. Some groups also use soft-lithography [16] to improve the pneumatic actuation. Paper, cloth, or fibers are embedded to elastomers to achieve asymetrical strain for ac-

tuation. Thus, the material loses flexibility, but becomes more resistant to high actuation pressures and force application.

C. Electronics

Electronics traditionally used in robotics are not compatible with soft robotics. We must therefore find a way to embed flexible sensors and electronics. This area is still evolving, but solutions have already been proposed. The most important of these is the inclusion of liquid metals between layers of lithographed polymers [7]. Such method could allow to embed sensors that can measure curvature [16], for example. Biological or chemical sensors would also be more adapted to the soft robotics field than traditionnal sensors.

Another challenge is the creation of soft power sources. Promising solutions for soft electrical power sources include graphene, organic polymers, or embedded conductive fabric [16].

III. Modeling and control

A. Modeling

If the movements of rigid robots can be broken down into simple movements in the plane, it is different for soft robots. The latter being able to bend, twist, extend, their kinematics become more complex. A better understanding of the movements of living beings such as octopuses [18] or caterpillars [2] has enabled the development of bio-inspired models for modeling soft robots. The fact that the deformations are continuous makes understanding the models difficult. In addition, soft robots are often underactuated.

It is therefore necessary to use sfying assumptions. One of them is the assumption of piecewise constant curvature [15], which consists of considering that the robot is an assembly of a finite number of curved joints. Bernoulli-Euler beam mechanics can then be used to predict movements. But this simplification does not allow all the specificities of soft robots to be taken into account, so non-constant curvature models have been developed [13].

A method commonly used in robotics is inverse kinematics [16], which makes it possible to calculate the position of all the links of a robot from the target position of a point on it. In the case of soft robotics, this method becomes more complex to use, as it is difficult to control the entire body of the robot. Solutions have been developed, using the PCC assumption. In [8], the approach is divided in two simpler task : move the effector of the robot to the desired position, and position the envelope of the robot in relation to the environment.

B. Control

Several solutions to design controllers have been proposed, two of them will be presented in this section.

1) Strain parametrization: In [11] is presented a geometric variable-strain (GVS) approach for static modeling of soft manipulators. The manipulator is modeled as a Cosserat rod [12](continuous stack of rigid cross-sections). In [19] is presented a work based on this model, to create a new implicit strain parametrization. A tip-pose and shape controller is designed on this parametrization, showing good results even with high order models.

2) Finite element method based controller: In [14] a dynamic control using finite element method (FEM) has been developed. The robot is first decomposed in finite elements. As it leads to a very large order Linear Time Invariant (LTI) system, the pole placement feedback method can't be used directly, therefore there is a need to reduce the order of the model. The model can be reduced using snapshot-Proper Orthogonal Decomposition (POD) [4]. As the number of parameters is lower, the pole placement method can be used to obtain the control matrix.

IV. Simulation

Several tools have been developed to facilitate the simulation of soft robots. We will focus here on three of the most recent ones.

A. SoRoSim

SoRosim is a MATLAB toolbox for modeling and simulation of soft robots. It uses the Geometric Variable Strains (GVS) approach to model the dynamic and static behavior of both rigid, soft or hybrid robots. This toolbox allows the user to create rigid or soft joints, assemble them, add external forces and actuators, and run the simulation.

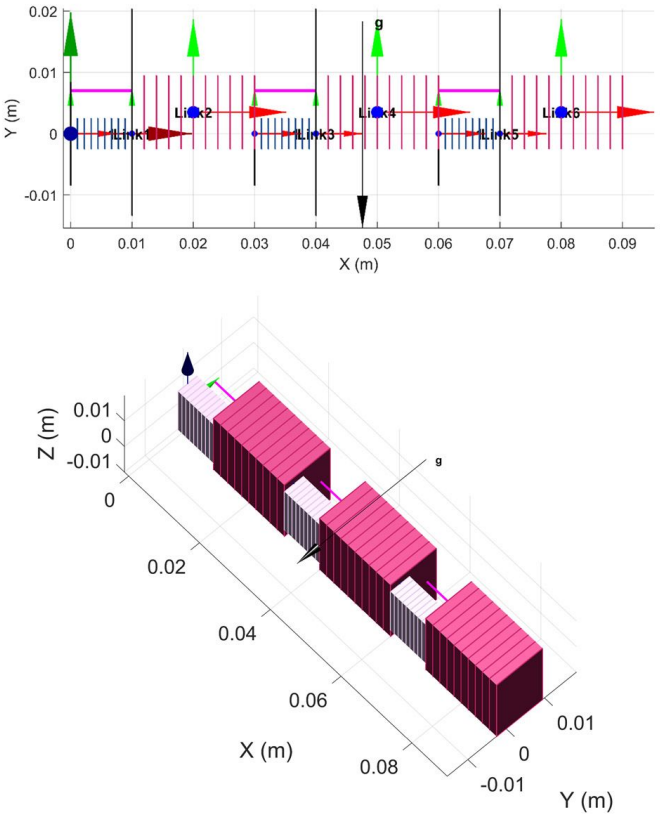


Fig. 1. Example of a soft finger in SoRoSim [23]

B. SOFA soft robotics toolkit

SOFA is an opensource framework designed for physics-based simulation. A plugin has been developed to enable interactive modeling and simulation of soft robots. It is based on the control model descrided in [3], [6]. Robots can be easily built using Python or C++ scripts, but also GUI.

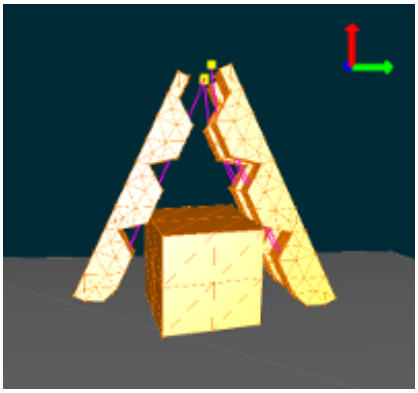


Fig. 2. Cable gripper simulated in SOFA [25]

C. SoMo

The opensource framework SoMo [20]is based on rigid-body physics, discretizing soft robots in multiple rigid elements. Robots can be built using Python scripts. Other plugins based on SoMo have been developed : SoMoGym [21], a plugin for training and reinforcement learning for soft robots controller, and SoMo-RL (built on SoMoGym), that allows for example to evaluate the influence of varying robot and control parameters on the training of RL policies.

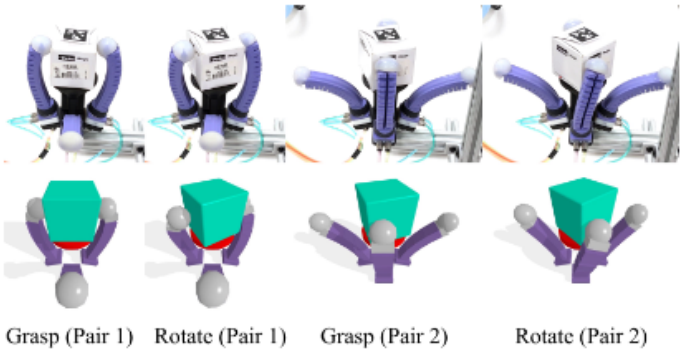


Fig. 3. Example of a manipulator simulation in SoMo [20]

V. Applications and systems

Soft robotics can have a wide range of applications, some of them will be presented in this part.

A. Medical applications

Rigid orthotics and medical devices can cause injury or discomfort due to their lack of flexibility. In addition, rigid connections sometimes do not allow the movements of organic limbs to be perfectly reproduced. Soft robotics is an interesting solution from this point of view, because the materials used have properties close to organic materials. Some recently developed systems are soft orthodics for human ankle-foot rehabilitation, soft sensing suits for lower limb measurement, and a soft system for simulation of cardiac actuation [16].

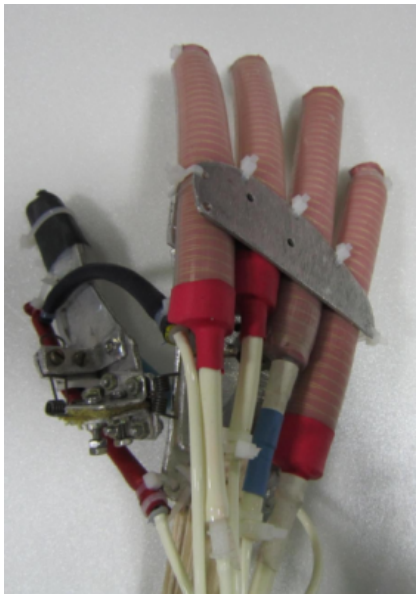


Fig. 4. Prosthetic hand designed by students during the Global Summer Immersion Programm in India

B. Objects manipulation

Gripping and manipulating objects is a very present problem in industry in particular. Although rigid robots are very common for this use, they do not allow them to adapt to a wide variety of objects. Soft robots, for their part, have a greater capacity for adaptation. Systems using iso-thermal stiffness variations, cable or pneumatic actuated fingers are highly efficient and adaptable [5], [16].

C. Locomotion

A lot of soft mobile robots have been developed using bio-inspired designs. Caterpillars, worms and octopuses [18], snakes, but also fishes [22] have been studied to help to understand and reproduce their behaviour. Some of them could be used to help in biological studies, as they achieve to have almost the same performances as their living counterpart.

One of the challenges in creating autonomous soft robots is the embedding of power sources, which are often quite heavy and therefore reduce the robot's loading capacity. One solution envisaged is to wire the soft robot to an unwired autonomous rigid robot, which will carry the power source [16].

VI. Conclusion

There are now very varied solutions concerning the design, simulation and control of soft robots, allowing them to be used in many fields. Although we have already been able to

Fig. 5. CSAIL's autonomous fish SoFI

Fig. 6. A chemically powered soft octopus [24]

identify a large number of materials that can be used for the manufacture of soft robots, such as polymers and gels, there is nevertheless much research to be done on the manufacture of soft electronics.

Modeling and controlling soft robots is complex due to material properties, but various solutions have been explored, such as segmenting robots into a large number of rigid connections, or using finite elements. These techniques also allow the simulation of robots, and have been implemented in several tools and several languages.

Soft robots are now used in many fields, whether in medicine, industry, or environmental studies for example.

Even if the list of technical solutions developed in this article is not exhaustive, it gives a good overview of the state of progress in soft robots. Many advances are still to be expected in the future, whether in the control, manufacturing or applications of soft robots.

REFERENCES

[1] Cheng, Nadia G., et al. "Design and Analysis of a Robust, Low-Cost, Highly Articulated Manipulator Enabled by Jamming of Granular Media." 2012 IEEE International Conference on Robotics and Automation, IEEE, 2012, pp. 4328–33, https://doi.org/10.1109/ICRA.2012.6225373.

[2] Coyle, Stephen, et al. "Bio-Inspired Soft Robotics: Material Selection, Actuation, and Design." Extreme Mechanics Letters, vol. 22, 2018, pp. 51–59, https://doi.org/10.1016/j.eml.2018.05.003.

[3] Duriez, Christian. "Control of Elastic Soft Robots Based on Real-Time Finite Element Method." ICRA, 2013.

[4] Goury, Olivier. Computational Time Savings in Multiscale Fracture Mechanics Using Model Order Reduction. 2015.

[5] Hesheng Wang, et al. "Visual Servo Control of Cable-Driven Soft Robotic Manipulator." 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2013, pp. 57–62, https://doi.org/10.1109/IROS.2013.6696332.

[6] Largilliere, Frederick, et al. "Real-Time Control of Soft-Robots Using Asynchronous Finite Element Modeling." ICRA, 2015.

[7] Lu, Nanshu, and Dae-Hyeong Kim. "Flexible and Stretchable Electronics Paving the Way for Soft Robotics." Soft Robotics, vol. 1, no. 1, Mar. 2014, pp. 53–62, https://doi.org/10.1089/soro.2013.0005.

[8] Marchese, Andrew D., et al. "Design and Control of a Soft and Continuously Deformable 2D Robotic Manipulation System." 2014 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2014, pp. 2189–96, https://doi.org/10.1109/ICRA.2014.6907161.

[9] Mosadegh, Bobak, et al. "Pneumatic Networks for Soft Robotics That Actuate Rapidly." Advanced Functional Materials, vol. 24, no. 15, 2014, pp. 2163–70, https://doi.org/10.1002/adfm.201303288.

[10] Petralia, Michael T., and Robert J. Wood. "Fabrication and Analysis of Dielectric-Elastomer Minimum-Energy Structures for Highly-Deformable Soft Robotic Systems." 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010, pp. 2357–63, https://doi.org/10.1109/IROS.2010.5652506.

[11] Renda, Federico, Costanza Armanini, et al. "A Geometric Variable-Strain Approach for Static Modeling of Soft Manipulators With Tendon and Fluidic Actuation." IEEE Robotics and Automation Letters, vol. 5, no. 3, 2020, pp. 4006–13, https://doi.org/10.1109/LRA.2020.2985620.

[12] Renda, Federico, Frederic Boyer, et al. "Discrete Cosserat Approach for Multisection Soft Manipulator Dynamics." IEEE Transactions on Robotics, vol. 34, no. 6, 2018, pp. 1518–33, https://doi.org/10.1109/TRO.2018.2868815.

[13] Renda, Federico, Michele Giorelli, et al. "Dynamic Model of a Multibending Soft Robot Arm Driven by Cables." IEEE Transactions on Robotics, vol. 30, no. 5, 2014, pp. 1109–22, https://doi.org/10.1109/TRO.2014.2325992.

[14] Thieffry, Maxime, et al. "Dynamic Control of Soft Robots." IFAC World Congress, 2017, https://hal.science/hal-01558844.

[15] Webster, Robert J., and Bryan A. Jones. "Design and Kinematic Modeling of Constant Curvature Continuum Robots: A Review." The International Journal of Robotics Research, vol. 29, no. 13, 2010, pp. 1661–83, https://doi.org/10.1177/0278364910368147.

[16] Rus, Daniela, and Michael T. Tolley. "Design, Fabrication and Control of Soft Robots." Nature, vol. 521, no. 7553, May 2015, pp. 467–75, https://doi.org/10.1038/nature14543.

[17] J. C. Case, E. L. White and R. K. Kramer, "Soft Material Characterization for Robotic Applications",Soft Robotics, vol. 2, no. 2, pp. 80-87, 2015.

[18] Calisti, M. et al. An octopus-bioinspired solution to movement and manipulation for soft robots. Bioinspiration & biomimetics 6, 036002 (2011)

[19] Renda, Federico, et al. "Dynamics and Control of Soft Robots With Implicit Strain Parametrization." IEEE Robotics and Automation Letters, vol. 9, no. 3, 2024, pp. 2782–89, https://doi.org/10.1109/LRA.2024.3360813.

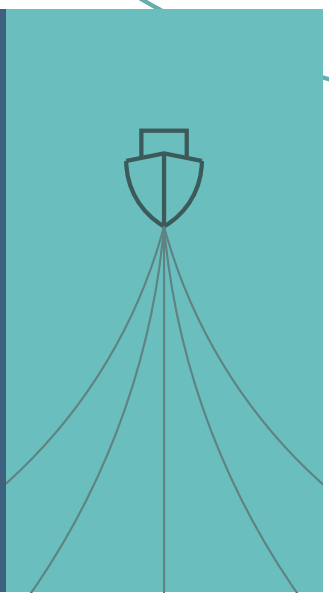
[20] Graule, Moritz A., Clark B. Teeple, et al. "SoMo: Fast and Accurate Simulations of Continuum Robots in Complex Environments." 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2021, pp. 3934–41, https://doi.org/10.1109/IROS51168.2021.9636059.

[21] Graule, Moritz A., Thomas P. McCarthy, et al. "SoMoGym: A Toolkit for Developing and Evaluating Controllers and Reinforcement Learning Algorithms for Soft Robots." IEEE Robotics

and Automation Letters, vol. 7, no. 2, 2022, pp. 4071–78, https://doi.org/10.1109/LRA.2022.3149580.
[22] Marchese, A. D., Onal, C. D. & Rus, D. Autonomous soft robotic fish capable of escape maneuvers using fluidic elastomer actuators. Soft Robotics 1, 75–87 (2014).
[23] Creating SoRoSim: A MATLAB Toolbox for Soft Robotics Modeling and Simulation. https://fr.mathworks.com/company/technical-articles/creating-sorosim-a-matlab-toolbox-for-soft-robotics-modeling-and-simulation.html.
[24] Shen, H. Beyond Terminator: squishy 'octobot' heralds new era of soft robotics. Nature (2016). https://doi.org/10.1038/nature.2016.20487
[25] Manti, Mariangela, et al. "An Under-Actuated and Adaptable Soft Robotic Gripper." Proceedings of the 4th International Conference on Biomimetic and Biohybrid Systems - Volume 9222, Springer-Verlag, 2015, pp. 64–74, https://doi.org/10.1007/978-3-319-22979-9_6.

28 février 2024

Transformer: A Novel Neural Network Architecture



Initiation à la Recherche 2024

MUSTIERE Ludovic

ENSTA BRETAGNE

Abstract

The recent advances in the field of Natural Language Processing (NLP) have allowed the development of large language models, such as the Generative Pre-trained Transformer (GPT) model. This model is based on the Transformer architecture, which is a neural network architecture that has been introduced in 2017. The Transformer architecture has been designed to solve the problem of machine translation, but it has been shown that it can be used for other tasks such as text generation.

Keywords

Neural Networks, Machine Learning, Transformer, Deep Learning, Natural Language Processing, Large Language Model, Generative Pre-trained Transformer.

Contents

1 Introduction

1.1 Context

Since early 2010s, Artificial Intelligence has known an unprecedented growth. Indeed, the progress in the field of Machine Learning (ML) has allowed to create algorithms capable of solving complex problems, such as image recognition or automatic translation. These algorithms are used in many fields, such as medicine, finance or robotics.

One of the fields where Artificial Intelligence has known the most success is Natural Language Processing. Indeed, Machine Learning algorithms have allowed to create models capable of understanding human language. These models are used in many fields, such as automatic translation, speech recognition or chatbots.

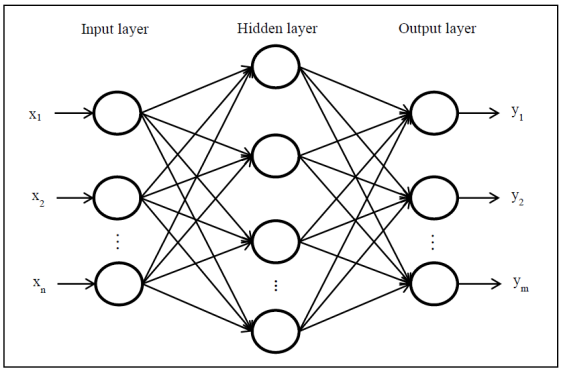
In this report, we will focus on how we can use Transformers to create an automatic translation algorithm. We will first explain what Transformers are, and how they work. Then we will explain how impactful they have been in the field of automatic translation.

2 Transformers

2.1 A quick introduction to neural networks

Before we can explain what Transformers are, we need to explain what neural networks are. Neural Networks (NN) are a set of algorithms that are designed to recognize patterns. They are inspired by the human brain, and they are composed of neurons. These neurons are organized in layers, and they are connected to each other. The first layer is called the input layer, and the last layer is called the output layer. The layers between the input and output layers are called hidden layers.

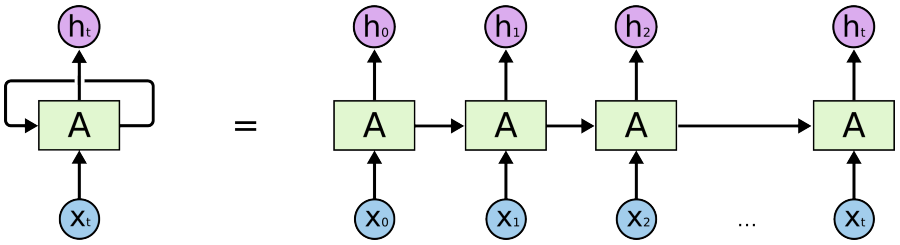
There are two broad types of neural networks: Feedforward Neural Networks (FNNs) and Recurrent Neural Networks (RNNs). FNNs are the simplest type of neural networks. The information moves in only one direction, forward from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network [Sch].



Source: Training back propagation neural networks using asexual reproduction optimization

Figure 1 – A Feedforward Neural Network

RNNs are designed to recognize a data's sequential characteristics and use patterns to predict the next likely scenario. The hidden layer in RNNs act as a memory, which allows them to remember previous inputs. This method is called bidirectional flow. Each input of that sequence has some relationship with its neighbors or has some influence on them. RNNs are used in many fields, such as handwriting recognition [GFS], language translation [SVL] or time series prediction [Pet].



Source: Colah's Blog

Figure 2 – A Recurrent Neural Network

However, RNNs have two major issues. Firstly, they are slow to train. Secondly, they suffer from short-term memory. Indeed, they are unable to remember information from the beginning of the sequence. This is called the vanishing gradient problem [BJZP].

To solve this problem, Long Short-Term Memory (LSTM)s were created. Long Short-Term Memorys [HS] are a special kind of RNNs that are capable of learning long-term dependencies. They include a branch that allows to retain information for later use. It improves the vanishing gradient problem but around 1000 words, it starts to forget the beginning of the sequence. Furthermore, they are even slower to train than RNNs. Finally, Long Short-Term Memorys take input sequentially one by one, which makes them difficult to parallelize.

2.2 Attention

An Attention mechanism allows the model to focus on the most relevant parts of the input sequence. Attention-based models are classified into two categories: *global* and *local* [LPM]. Common to these two types of models is the fact that they first take as input the hidden state h_t at each time step t. Then, they compute a context vector c_t that captures relevant source-side information for each target word. The concatenation of the hidden state h_t and the context vector c_t is used to compute the attentional vector $\tilde{h}_t$. Finally, the attentional vector is used to predict the target word. The difference between the two types of models is in the way they compute the context vector. Two popular attention-based models are the *Bahdanau* and *Luong* models [BCB, LPM]. You can find a possible way to compute the output in the equations (1) and (2) with $p(y_t|y_{<t}, x)$ being the probability of the target word y_t given the previous target words $y_{<t}$ and the source sentence x and W_c and W_s being the weight matrices of the context and softmax layers respectively that are learned during training:

$$\tilde{h}_t = \tanh(W_c[h_t; c_t]) \tag{1}$$

$$p(y_t|y_{<t}, x) = \text{softmax}(W_s\tilde{h}_t) \tag{2}$$

2.2.1 Global Attention

The global attention model considers all the hidden states of the source sentence to compute the context vector. In order to do that, we create a variable-length alignment vector a_t, whose length is equal to the length of the source sentence. The alignment vector is computed using the equations (3) and (4) which compares the current hidden state h_t with each source hidden state $\bar{h}_s$ [LPM]:

$$a_t(s) = \text{align}(h_t, \bar{h}_s) \tag{3}$$

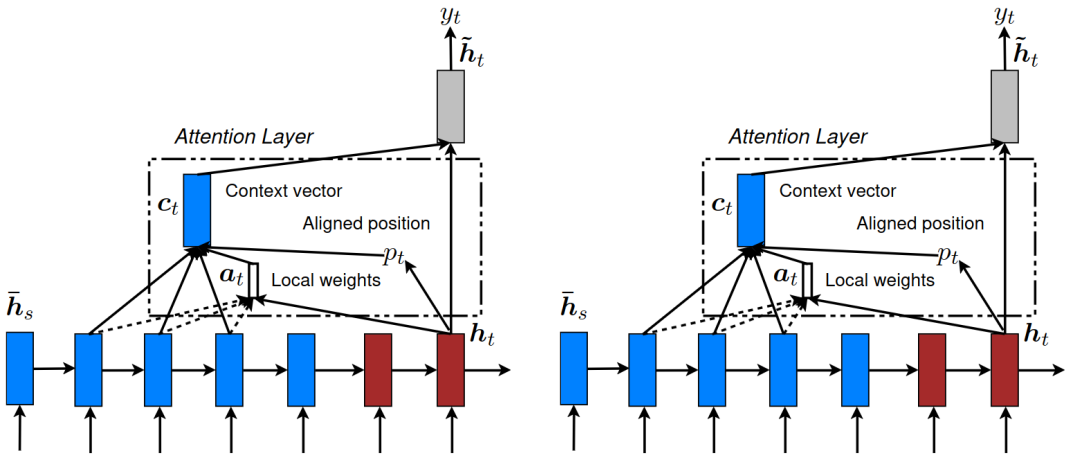
$$a_t(s) = \frac{\exp(\text{score}(h_t, \bar{h}_s))}{\sum_{s'} \exp(\text{score}(h_t, \bar{h}_{s'}))} \tag{4}$$

The score function can be computed using different methods. One of the most popular methods is the dot product between the current hidden state and the source hidden state $score(h_t, \bar{h}_s) = h_t^T \bar{h}_s$. The context vector c_t is then computed using the alignment vector a_t and the source hidden states $\bar{h}_s$ [LPM]:

$$c_t = \sum_s a_t(s)\bar{h}_s \tag{5}$$

2.2.2 Local Attention

The local attention model, on the other hand, only considers a subset of the source hidden states to compute the context vector. The subset is determined by a window centered around the current target position. In concrete details, the model generates an



(a) Global attention model (b) Local attention model

Figure 3 – Attention mechanism

aligned position p_t for each target position t. The aligned position can either be assumed to be equal to t in the case of a *monotonic alignment* assuming that the target position is aligned with the source position or be predicted by the model in the case of a *predictive alignment*. The equation (6) shows how the aligned position is computed in the case of a predictive alignment with v_p and W_p being learned through training and S being the source sentence length [LPM]:

$$p_t = S \cdot \text{sigmoid}(v_p^T \tanh(W_p h_t)) \tag{6}$$

The context vector c_t is then computed as the weighted average over the set of source hidden states $\bar{h}_s$ within the window $[p_t - D, p_t + D]$ with D being the window size empirically selected. The alignment vector a_t is computed to favor alignment points near the aligned position p_t using the equation (7) with σ being the standard deviation of the Gaussian distribution [LPM]:

$$a_t(s) = align(h_t, \bar{h}_s) \cdot \exp(-\frac{(s - p_t)^2}{2\sigma^2}) \tag{7}$$

2.3 The Transformer

The Transformer is a novel neural network architecture that was introduced in the paper "Attention is All You Need" [VSP+]. It is based on the attention mechanism and it is designed to solve the vanishing gradient problem. The goal is to reduce the number of sequential operations and to train the model faster than Long Short-Term Memorys. The main idea is to remove the use of RNNs or Convolutional Neural Networks (CNNs) and to use only self-attention layers. In the previous models like Conv2Seq [GAG+] or ByteNet [KES+], the input sequence is passed through a stack of convolutional layers to

extract local features. Then, the output of the convolutional layers is passed through a stack of RNNs to extract global features. The number of operations required to relate signals from two arbitrary input or output positions grows in the distance between positions, linearly for Conv2Seq and logarithmically for ByteNet. In the Transformer, this is reduced to a constant number of operations, improving the training speed and the performance of the model.

The Transformer is composed of two main parts: the encoder and the decoder. The encoder maps an input sequence of symbol representations $(x_1, x_2, ..., x_n)$ to a sequence of continuous representations $(z_1, z_2, ..., z_n)$. Given this vector representation, the decoder then generates an output sequence of symbols $(y_1, y_2, ..., y_m)$ one element at a time. It continuously uses the previously generated elements to generate the next one.

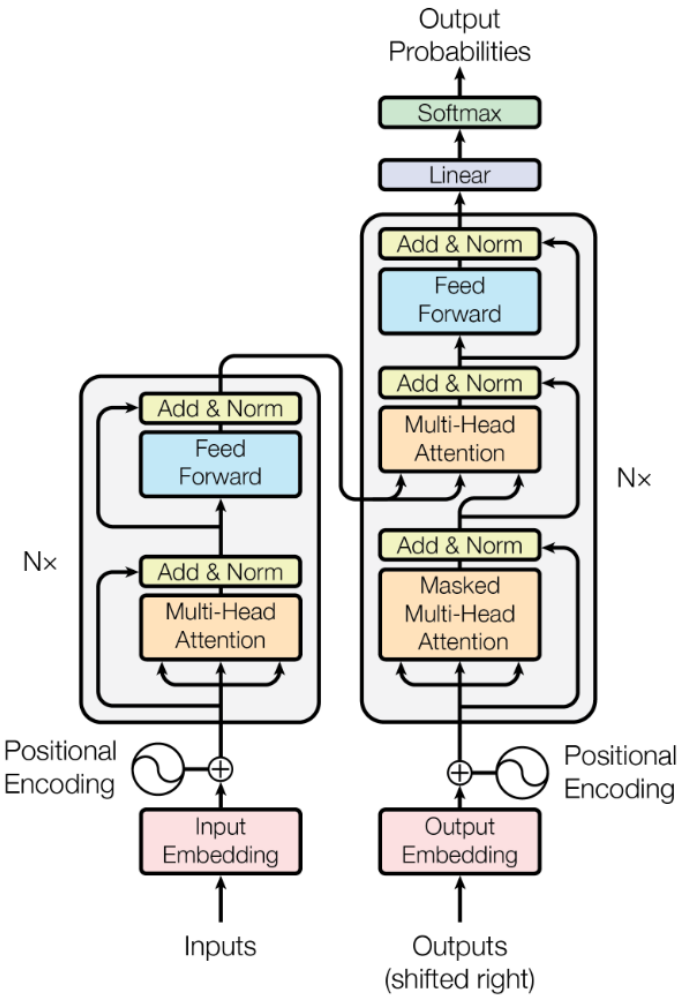


Figure 4 – The Transformer architecture

If you want to train a translator for English to French, we give the English sentence to the encoder and the French sentence to the decoder.

2.3.1 Encoder block

The encoder is composed of an embedding layer, a positional encoding layer and a stack of N identical layers. The input sequence is first passed through the embedding layer to convert the input tokens (words) into continuous representations. The idea is to assign to every word a unique vector representation and to keep words with similar meanings close to each other. Then, the positional encoding layer is used to add information about the position of the word in the sequence in order to give context to the model. Once the input sequence is embedded and positional encoded, it is passed through the Multi-Head Attention layer. For every word in the sequence, the model computes multiple attention vectors per word in parallel in order to capture different interactions with the other words in the sequence. Then, it takes a weighted average of the attention vectors to get the final attention vector of every word. The output of the Multi-Head Attention layer is passed through a simple position-wise fully connected feed-forward network. Since every attention vector is independent of the others, the model can parallelize the computation of the attention vectors, unlike in RNNs or Long Short-Term Memorys.

2.3.2 Decoder block

The decoder is composed of an embedding layer, a positional encoding layer, a stack of N identical layers and a linear layer. The Masked Multi-Head Attention layer is used to prevent the model from looking at the future words in the sequence, which is not possible in a real-world scenario. To explain the need of such a layer, we need to understand how the model learns. When we provide an English sentence to the encoder, it will be translated to a French sentence using only previous results. The model then compares the output with the expected French sentence and updates the weights of the model. If the model looks at the future words in the sequence, it will be able to cheat and to generate the correct translation without learning anything. So, we need to make the model look only at the i first words in the sequence to generate the i-th word. Now, the output of the Masked Multi-Head Attention layer is passed through the Multi-Head Self-Attention layer with the encoder output. The idea is to make the mapping between the English and French words and to capture the interactions between the words in the two sequences. The output is an attention vector representing the relationship with other words in both languages. Finally, the attention vector is passed through a simple position-wise fully connected feed-forward network and a linear layer to generate the output sequence. The choosen word is the one with the highest probability.

Now we will do a deep dive into each layer of the Transformer. In this work, $N = 6$, $d_{model} = 512$, $h = 8$, $d_k = d_v = d_{model}/h$ [VSP+].

2.3.3 Positional encoding

As explained earlier, the positional encoding layer is used to add information about the position of the word in the sequence. The positional encoding have the same dimension d_model as the word embeddings so they can be summed. There are many ways to compute the positional encoding. The authors of the paper used the formulas (8) and (9) where pos is the position of the word in the sequence, i is the dimension of the positional

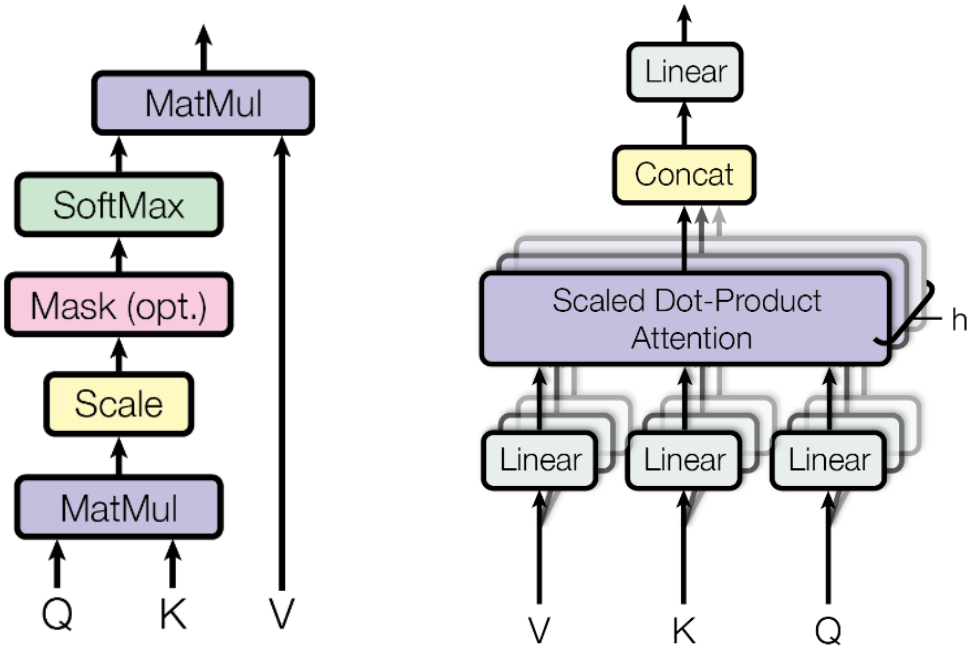
encoding. The authors chose this method because it allows the model to easily learn to attend by relative positions, which is important for translation, since for any offset k, PE_{pos+k} can be represented as a linear function of PE_{pos}. This also allows the models to extrapolate to sequence lengths longer than the ones encountered during training.

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{8}$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{9}$$

2.3.4 Multi-Head Attention

The Multi-Head Attention layer is used to compute multiple attention vectors per word in parallel. It uses multiple layers of Scaled Dot-Product Attention to compute simultaneous attention vectors. The Scaled Dot-Product Attention is computed using the equation (10). The input is first passed through three linear layers to create the query, key and value vectors, respectively of size d_k, d_k and d_v. The query and key vectors are then multiplied together to get the attention score, each divided by $\sqrt{d_k}$ to prevent the vanishing gradient problem. The attention score is then passed through the softmax function to get the attention weights. The queries, keys and values are in practice packed together in to matrices Q, K and V.



(a) Scaled Dot-Product Attention (b) Multi-Head Attention

Figure 5 – Multi-Head Attention

Multi-Head Attention then project the queries, keys and values h times with different,

learned linear projections to d_k, d_k and d_v dimensions. The attention scores are then computed in parallel and concatenated. The concatenated attention scores are then passed through a final linear layer to get the final attention vector. The Multi-Head Attention layer is computed using the equations (10), (11) and (12).

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{10}$$

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{11}$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, ..., \text{head}_h)W^O \tag{12}$$

2.3.5 Position-wise Feed-Forward Networks

The Position-wise Feed-Forward Networks layer consist of two linear transformations with a ReLU activation in between (two convolutional layers with kernel size 1). The input is first passed through the first linear layer to get the intermediate representation. Then, the intermediate representation is passed through the ReLU activation function to get the final representation. The final representation is then passed through the second linear layer to get the output of the layer. The Position-wise Feed-Forward Networks layer is used to capture the interactions between the words in the sequence. The inner dimension of the Position-wise Feed-Forward Networks layer is $d_{ff} = 2048$.

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{13}$$

3 Training and Results

The model was trained on the WMT 2014 English-French dataset of about 36M sentences and 32000 words. The model was trained for 100,000 steps with a batch size of 25000 tokens on 8 NVIDIA P100 GPUs. Each training step took about 0.4 seconds for a total of 12 hours. Another model named *big* was trained on the same dataset for 300000 steps. Each training step took about 1.0 seconds for a total of 3.5 days.

The results of the model were compared to the results of the previous state-of-the-art models. The model achieved a BLEU score of 41.8 on the newstest2014 dataset, which is an improvement of 2.2 BLEU points over the previous state-of-the-art model for a reduced training cost [VSP+].

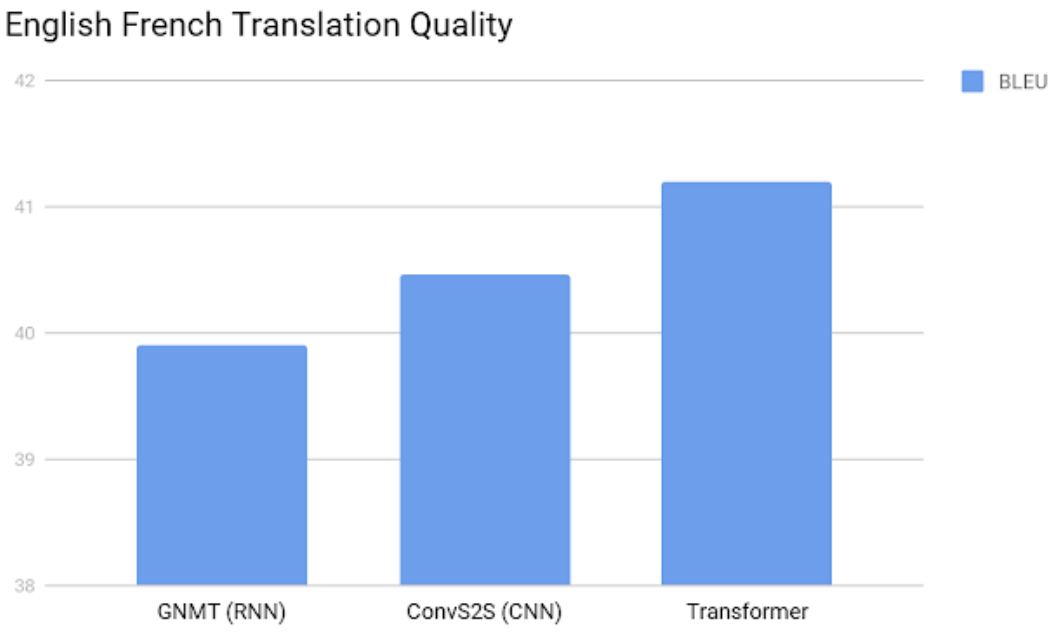


Figure 6 – Results of the Transformer model

4 Conclusion

In this report, we have explained what Transformers are, and how they work. We have also explained how impactful they have been in the field of automatic translation. We have seen that the Transformers model achieved a BLEU score of 41.8 on the newstest2014 dataset, which is an improvement of 2.2 BLEU points over the previous state-of-the-art model for a reduced training cost. We have also seen that the Transformers model is faster to train than Long Short-Term Memorys and RNNs. The Transformer model is now widely used in the field of automatic translation, and it has allowed to create more accurate and faster automatic translation algorithms but also in the field of chatbots and speech recognition. Soon, it may be widely used in the field of medicine, finance or robotics.

For additional information, you can check Jay Alammar's blog amazing Github repository Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models With Attention). It contains a lot of visualizations and explanations about the Transformer model and the attention mechanism.

List of Figures

Glossary

Feedforward Neural Network A feedforward neural network is an artificial neural network wherein connections between the nodes do not form a cycle. As such, it is different from its descendant: recurrent neural networks. The feedforward neural network was the first and simplest type of artificial neural network devised. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network. 1

Generative Pre-trained Transformer Generative Pre-trained Transformer (GPT) is a large language model introduced in 2018. This model has been trained on a large corpus of text, and it can be used to generate text. i

Long Short-Term Memory Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. It can not only process single data points (such as images), but also entire sequences of data (such as speech or video). For example, LSTM is applicable to tasks such as unsegmented, connected handwriting recognition, speech recognition and anomaly detection in network traffic or IDSs (intrusion detection systems). 2

Machine Learning Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves. 1

Natural Language Processing Natural language processing (NLP) is a branch of artificial intelligence that helps computers understand, interpret and manipulate human language. NLP draws from many disciplines, including computer science and computational linguistics, in its pursuit to fill the gap between human communication and computer understanding. i, 1, 4

Neural Networks A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. Neural networks can adapt to changing input; so the network generates the best possible result without needing to redesign the output criteria.. 1

Recurrent Neural Network A recurrent neural network (RNN) is a type of artificial neural network commonly used in speech recognition and natural language processing (NLP). RNNs are designed to recognize a data's sequential characteristics and use patterns to predict the next likely scenario. 1, 2, 4

Transformer The Transformer is a deep learning model introduced in 2017, used primarily in the field of Natural Language Processing. Like RNNs, Transformers are designed to handle sequential data, such as natural language, for tasks such as translation and text summarization. However, unlike RNNs, Transformers do not require that the sequential data be processed in order. i, 1

References

[BCB] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. 3

[BJZP] Sunitha Basodi, Chunyan Ji, Haiping Zhang, and Yi Pan. Gradient amplification: An efficient way to train deep neural networks. 3(3):196–207. 2

[GAG+] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional Sequence to Sequence Learning. 4

[GFS] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. Multi-dimensional Recurrent Neural Networks. In Joaquim Marques De Sá, Luís A. Alexandre, Włodzisław Duch, and Danilo Mandic, editors, *Artificial Neural Networks – ICANN 2007*, volume 4668, pages 549–558. Springer Berlin Heidelberg. 2

[HS] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-term Memory. 9:1735–80. 2

[KES+] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. Neural Machine Translation in Linear Time. 4

[LPM] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective Approaches to Attention-based Neural Machine Translation. 3, 4

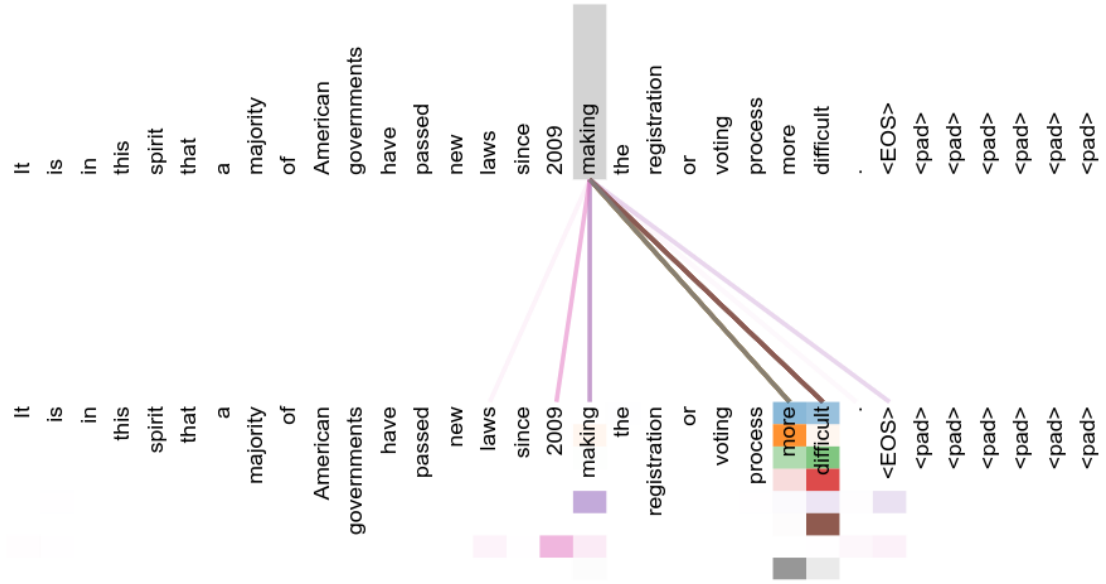
[Pet] Gábor Petneházi. Recurrent Neural Networks for Time Series Forecasting. 2

[Sch] Juergen Schmidhuber. Deep Learning in Neural Networks: An Overview. 61:85–117. 1

[SVL] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to Sequence Learning with Neural Networks. 2

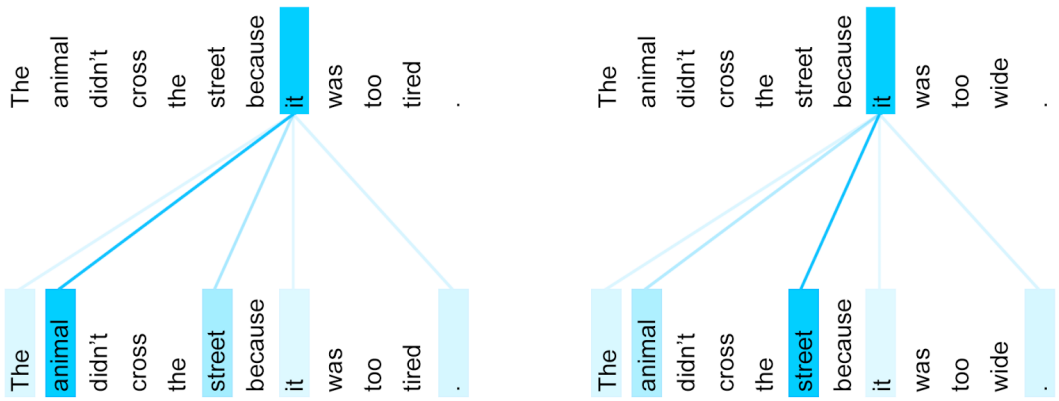
[VSP+] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. 4, 6, 8

A Annexes



Source: Attention is All You Need

Figure 7 – An example of the attention mechanism following long-distance dependencies in the encoder self-attention in layer 5 of 6. Many of the attention heads attend to a distant dependency of the verb ḿaking; completing the phrase ḿaking...more difficult; Attentions here shown only for the word ḿaking; Different colors represent different heads. Best viewed in color.



Source: Transformer: A Novel Neural Network Architecture for Language Understanding

Figure 8 – The encoder self-attention distribution for the word “it” from the 5th to the 6th layer of a Transformer trained on English to French translation (one of eight attention heads).

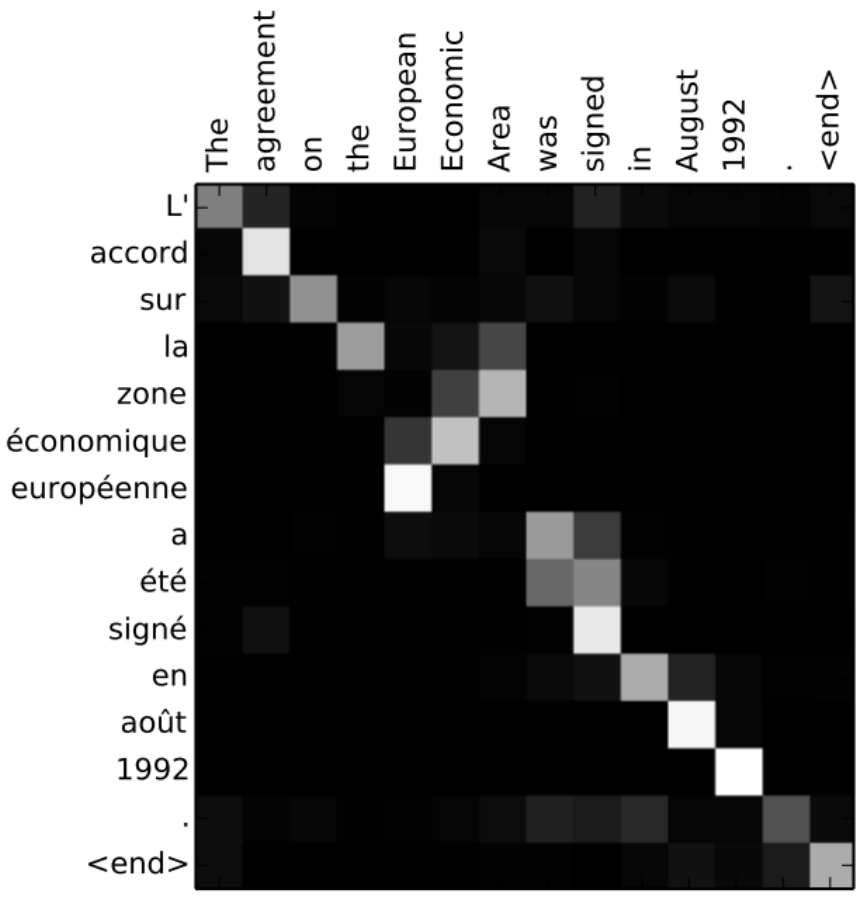


Figure 9 – You can see how the model paid attention correctly when outputing "European Economic Area". In French, the order of these words is reversed ("européenne économique zone") as compared to English. Every other word in the sentence is in similar order.

Sound Source Localization Using UAV Swarm, With Interval Calculation Methods

Virgile Pelle
ENSTA Bretagne, Lab-STICC, UMR CNRS 6285, Brest, France

***Abstract*—This paper addresses the localization of a sound source through the application of a swarm of unmanned aerial vehicles (UAVs). Given the unknown emission time of the sound signal, our methodology relies solely on the difference of the time-of-arrival at each UAV. In order to take the time and position measurements errors into account, we will employ interval calculation methods.**

***Index Terms*—sound source localization, time difference of arrival (TDOA), interval analysis**

I. Introduction

Sound source location research is an active area of research. Indeed, such a methodology can be very useful for finding victims in natural disasters, where rescue teams are deployed on the ground to find them. We can imagine a swarm of lightweight UAVs flying over the research zone trying to hear a sound such as a whistle to locate the victims.

The challenge in this scenario is that the emission time of the sound is unknown, constituting what is known as a Time-Difference-Of-Arrival (TDOA) localization problem. Each drones captures the sound at a distinct moment. Then by communicating between them, they will share both their spatial coordinates and the time at which they receive the sound signal. And finally, given all the information they have, the drones collaborate using interval analysis methods to deduce the location of the sound source.

For the purpose of this study, we make the assumption that there is only one immobile source to locate. The research is conducted with the utilization of four UAVs. These UAVs will maintain stationary flight and their relative position between each other will be known. Additionally, they all share a common clock, ensuring synchronized time measurements between them.

This article focuses on the formulation of the localization equation and the implementation of a constraint network designed to solve the localization problem.

II. Related Work

In the context of source localization using drone swarms, and the unique challenges posed by unknown transmission time, it is pertinent to review existing literature.

Time-based techniques are an important category. Methods such as time of flight (TOA) [2] [7] and time difference of arrival (TDOA) [3] exploit the temporal aspects of signal propagation between receivers and a source, enabling accurate distance determination. In addition, the angle-of-arrival (AOA) approach [4] has been proposed, requiring an array of receiver antennas to estimate position based on signal angles. There is also other methods employed for localization of source such as Received Signal Strength Indicator (RSSI) [1] for indoor localization, or also by combining it with ultrasonic sounds [5].

In the domain of sound source localization and UAV swarm applications, notable research has been conducted to optimize UAV positions TDOA applications [6]. This work explores optimal positions for a swarm of 4 UAVs in TDOA scenarios, for more accurate sound source localization.

III. Theory

Let's consider the source we want to find is located in a 3D space at $s(x_s, y_s, z_s) \in \mathbb{R}^3$. We have a swarm of 4 UAVs $Q = [p1, p2, p3, p4] \in \mathbb{R}^{3\times 4}$.

Given that there is no time synchronisation between the source and the UAVs, we have to rely on the difference in time between the reception on the sound at each drones. Let's define t_i the time of arrival at p_i. We can know define the difference in distance from source to drone p_i and source to drone p_j.

$$\begin{aligned} d_{ij} &= c(t_i - t_j) \\ d_{ij} &= -d_{ij} \end{aligned} \tag{1}$$

where c is the speed of sound.

We can also express d_{ij} using the distances between the source to drones

$$d_{ij} = D_i - D_j \tag{2}$$

where

$$D_i^2 = (x_i - x_s)^2 + (y_i - y_s)^2 + (z_i - z_s)^2, i \in [1, 4] \tag{3}$$

where (x_1, y_1, z_1) is the position of drone n°1 (similarly for drones 2,3 and 4).

Expandings (3) and using D_2,D_3,D_4 given by (2) yields

$$\begin{aligned} D1^2 = x_1^2 - 2x_1x_s + x_s^2 + y_1^2 - 2y_1y_s + x_s^2 \\ +z_1^2 - 2z_1z_s + x_s^2 \end{aligned} \tag{4}$$

$$\begin{aligned} (d_{i1} + D_1)^2 = x_i^2 - 2x_ix_s + x_s^2 + y_i^2 - 2y_iy_s + y_s^2 \\ +z_i^2 - 2z_iz_s + z_s^2 \end{aligned} \tag{5}$$

Let's define the squared radius to the source and to each drones:

$$\begin{aligned} R_s &= x_s^2 + y_s^2 + z_s^2 \\ R_i &= x_i^2 + y_i^2 + z_i^2, i \in [1, 4] \end{aligned}$$

Using (4) we can rewrite R_s as follow:

$$R_s^2 = -R_1 + D_1^2 + 2x_1x_s + 2y_1y_s + 2z_1z_s \tag{6}$$

Substituting (6) into (5) yields:

$$\begin{aligned}(d_{i1} + D_1)^2 - R_i^2 + R_1^2 - D_1^2 = 2x_1x_s + 2y_1y_s + 2z_1z_s \\ -2x_ix_s - 2y_iy_s - 2z_iz_s, i \in [2,4]\end{aligned} \tag{7}$$

We can choose to take drone n°1 as origin. This makes : $(x_1, y_1, z_1) = (0, 0, 0)$ and $R_1 = 0$ which simplify (7):

$$R_i^2 - d_{i1}^2 - 2d_{i1}^2 D_1 = 2x_ix_s + 2y_iy_s + 2z_iz_s, i \in [2,4] \tag{8}$$

(4) can also be simplified and gives $D_1 = R_S$. Thus 8 can be written as:

$$\Delta - 2R_sd = 2M \begin{pmatrix} x_s \\ y_s \\ z_s \end{pmatrix} \tag{9}$$

where

$$\Delta = \begin{pmatrix} R_2^2 - d_{21}^2 \\ R_3^2 - d_{31}^2 \\ R_4^2 - d_{41}^2 \end{pmatrix}, d = \begin{pmatrix} d_{21} \\ d_{31} \\ d_{41} \end{pmatrix}, M = \begin{pmatrix} x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \\ x_4 & y_4 & z_4 \end{pmatrix},$$

If we choose p_2 p_3 and p_4 in order to have a linearly independent family of vectors, then M is inversible and we can thus write:

$$s = \begin{pmatrix} x_s \\ y_s \\ z_s \end{pmatrix} = \frac{1}{2} M^{-1}(\Delta - 2R_sd) \tag{10}$$

The equation (10) still involve the unknown source radius R_s. To solve this recall that:

$$R_s = (s^T s)^{\frac{1}{2}} \tag{11}$$

By substituting in (10) we obtain the following quadratic equation and its solution:

$$aR_s^2 + bR_s + c = 0 \tag{12}$$

$$R_s = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}, R_s > 0 \tag{13}$$

where

$$\begin{aligned} a &= 4 - 4d^T(M^{-1})^T M^{-1} d \\ b &= 2d^T(M^{-1})^T M^{-1}\Delta + 2\Delta^T(M^{-1})^T M^{-1} d \\ c &= -\Delta^T(M^{-1})^T M^{-1}\Delta \end{aligned}$$

In order to solve the whole problem, equation (13) is solved in a first time to obtain the source radius and then knowing this radius equation (10) gives us the source position.

In some scenario, (13) will returns two physical solution. In these cases two source location will be identified. When it happens the two location are usually far apart and the correct location can be determined by other physical reasoning for example if a solution is found outside the research zone.

IV. Interval Analysis

In order to perform interval resolution of our problem we need to define a sets of contractors.

$$\begin{cases} (i) & d_{ij} = c(t_i - t_j) \\ (ii) & \|p_i\|^2 = R_i^2 \\ (iii) & R_i^2 - d_i 1^2 - 2R_s d = 2p_i^T s \end{cases}$$

- *(i)* evaluates the difference of distance from source to drone p_i and source to drone p_j.
- *(ii)* evaluates the distance from origin to drone p_i
- *(iii)* evaluates the source position given the information of drone p_i. This is the main contractor derived from equation (9)

The solution of the equation (13) to estimate R_s can not be solved using interval analysis due to its complexity. Apart from this equation, everything else is computed using interval analysis and still provide an interval approach for the final solution of the problem.

V. Application

Let's consider a swarm of 4 UAVs flying in an Y-shape with a baseline of 50m as shown on *Figure 1*. The drone n°1 is flying 10 meters above the other to avoid them being all in the same plan. The others 3 are flying at 10m above the ground.

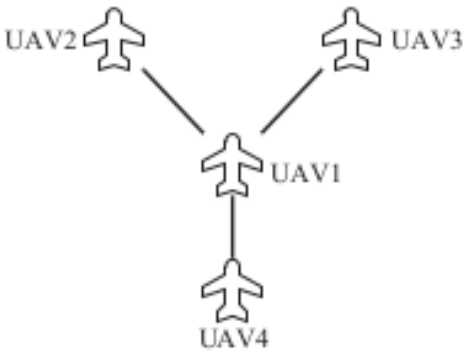


Fig. 1. Flying layout

The position and the time of reception of the sound for each drone are measured. We will add an uncertainty of $\pm$ 1 meter for the position and $\pm$ 10ms for the time of arrival.

We will only focus on the result in the 2D plan because we assume that the source is located on the ground.

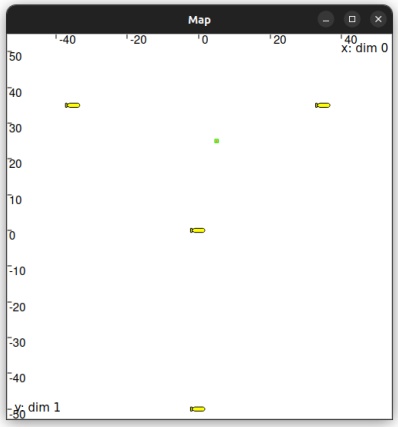


Fig. 2. Ideal conditions simulation

The simulation in *Figure 2* assumed precise position and time measurements without introducing noise. The source location is determined exactly (the little green dot), showcasing the robustness of the proposed methodology under ideal conditions.

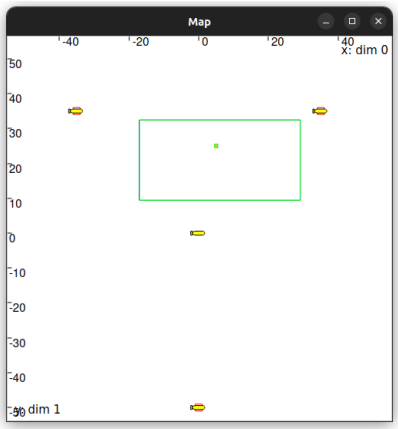


Fig. 3. Simulation with uncertainty (flying over)

As soon as the sensors uncertainty are taken into account, the solution become much less precise. Those results are highlighted in the *Figure 3*. The interval calculation computes a solution interval which contains the real position of the source but with a lot of uncertainty. In this simulation the surface of the solution interval measures around 1000 squared meters.

It is important to note that in these situation the 4 UAVs are flying above the source.

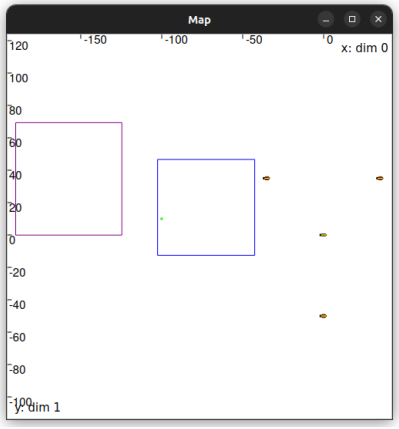


Fig. 4. Simulation with uncertainty (flying away)

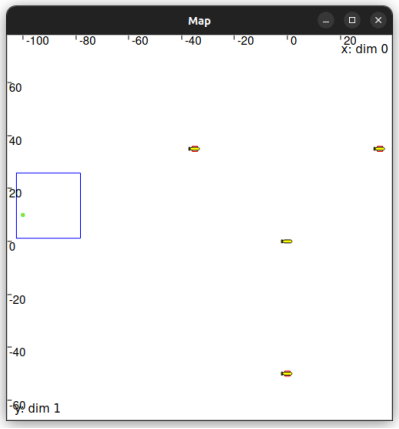


Fig. 5. Simulation with reduced uncertainty (flying away

When the source is far away from the swarm *Figure 4* the algorithm returns 2 intervals and only one of them cover the real position of the source. The main reason is that equation 13 returns 2 acceptable solutions because the sensors uncertainty are included. Indeed as shown in the *Figure 5* where the time uncertainty has been lowered to $\pm$ 1ms, the algorithm finds again only one interval and its size is about 600 squared meters.

VI. Conclusion

This paper introduced a method for sound source localization employing a UAV swarm in scenarios where emission time was unknown. By utilizing time differences of arrival and incorporating interval analysis to handle measurement uncertainties, the proposed approach demonstrated promising results in three-dimensional space localization.

This work contributes to advancing sound source localization using UAV swarms, with potential applications in disaster response.

Further work including a moving swarm above the whole research zone and the ability of handling multiple sound detection, would enhance the robustness of this method.

LIBRARIES

The CODAC is a C++/Python library providing tools to guarantee computations over intervals of reals, sets and trajectories. The official web page is www.codac.io .

ACKNOWLEDGMENTS

We thank S. Rohou for the developement of the CODAC library.

REFERENCES

[1] P. Bahl and V.N. Padmanabhan. Radar: an in-building rf-based user location and tracking system. 2:775–784 vol.2, 2000.

[2] Shibkali Bera, Sanjoy Kumar Mondal, and Pampa Sadhukhan. Evaluation of toa-based localization schemes using range estimation at network layer in wlan environment. pages 1–6, 2011.

[3] Hyuntae Cho, Yeonsu Jung, Hoon Choi, Hyunsung Jang, Sanghyun Son, and Yunju Baek. Precise location tracking system based on time difference of arrival over lr-wpan. pages 67–72, 09 2008.

[4] D. Niculescu and Badri Nath. Ad hoc positioning system (aps) using aoa. 3:1734–1743 vol.3, 2003.

[5] Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan. The cricket location-support system. page 32–43, 2000.

[6] Zhou Ronghua, Sun Hemin, Li Hao, and Luo Weilin. Tdoa and track optimization of uav swarm based on d-optimality. *Journal of Systems Engineering and Electronics*, 31(6):1140–1151, 2020.

[7] Yimao Sun, K.C. Ho, Yanbing Yang, Lei Zhang, and Liangyin Chen. Computationally attractive and statistically efficient estimator for noise resilient toa localization. *Signal Processing*, 200:108663, 2022.

The impact of image preprocessing on the creation of an optimized machine learning model for identifying the position of a goal frame.

Martin Pilon

ENSTA Bretagne - Autonomous Robotics
martin.pilon@ensta-bretagne.fr

Abstract

Optimizing a machine learning model for visual servoing and identifying the position of a red goalpost, in the context of NAO football, involves crucial steps in image preprocessing. Techniques such as applying a threshold on the red color (thresholding) to binarize the image and reducing its size are essential for improving the quality of visual data. Thresholding helps segment the image, isolating the goalpost and reducing noise, while size reduction decreases complexity while preserving essential features. This preprocessing approach, tailored to the specific nature of red goalposts, enhances the model's ability to extract relevant information, leading to improved visual servoing performance. This study emphasizes the positive impact of such preprocessing methods on the accuracy of the machine learning model for red goalpost detection.

1 Introduction

The framework of our mission is to score a goal using a NAO robot. The goalpost depicted below must be detected and positioned to enable the NAO robot to be servoed relative to it. To detect and position the goalpost, possible methods include machine learning (ML) approaches. ML methods require a database for the model to learn from.

Figure 1: NAO and goalpost

This article focuses on optimizing the input of a model dedicated to goalpost detection. The input of the model is a image. Assuming we are in a scenario involving embedded machine learning, and thus, we need to have the most efficient model possible. The challenge is that, as we aim for the simplest model to avoid underfitting, we want the input parameters of our model to be as straightforward as possible [1].

The emphasis is on data preprocessing (DP), a crucial phase to ensure the accuracy and robustness of the model. Specifically, we delve into techniques such as Thresholding for image binarization and size reduction. These processes play a crucial role in improving the quality of visual data and reducing the amount of information, essential aspects for achieving optimal model performance.

2 Current knowledge

We have a fixed input dataset determined by measurements from various sensors. Data preprocessing is a crucial step in ML training, and several criteria can be applied to optimize our input dataset [2]:

- Handling missing values: Completing missing values is essential, as data often come from sensors with a measurement frequency. Some models, however, require values of interest that have not been measured.
- Reducing the size of the dataset: This can be done through two methods. Data preprocessing can either remove elements from the data that are not exploitable or not very interesting, or it can decrease the data resolution. Reducing the size of the data helps eliminate (or decrease) anomalies and focuses the model training on areas of interest.
- Increasing the size of the dataset: For example, in images, this is done by changing the scale and orientation of the image. This increases the dataset size and improves the model's generalization.
- Improving the quality of the data: Applying filters or functions to a dataset can highlight data that is interesting for ML exploitation.

To achieve this, data preprocessing can use a variety of methods to improve the quality of the input datasets. In our case, the preprocessing addresses two criteria: reducing the size of the dataset and improving the quality of the data. The input is an image. Therefore, we can use image preprocessing methods to enhance the quality of visual data before employing them in machine learning models. There are numerous methods employed in image preprocessing, and we will primarily use resizing and thresholding [3].

3 Theory

3.1 Tresholdind

Thresholding method involves taking a color image and converting it into a binary layer that adheres to the following conditions for each pixel in an image [4]:

$$I > T$$

- I : Intensity of the pixel
- T : Threshold

If this condition is satisfied, then the pixel of the corresponding layer will be 1, otherwise 0. Using a threshold allows converting an image where each pixel has $256 * 256 * 256 = 16777216$ values (corresponding to its RGB) to only 2 values per pixel (1 or 0).

In our case, what we desire from our thresholding is to return 1 if the pixel is red and 0 otherwise. This method will help extract our object from the image (assuming

that the background image does not contain red). This will enable the extraction of important data for machine learning, thus improving the quality of our dataset while reducing the amount of information. To achieve this on an Hue Saturation Value (HSV) light model, we can set the following condition:

$$Hmax > H > Hmin$$

$$Smax > S > Smin$$

$$Vmax > V > Vmin$$

- H: Hue value of the pixel
- S: Saturation value of the pixel
- V: Value (brightness) value of the pixel
- $Hmin$: Predefined minimum threshold for Hue
- $Hmax$: Predefined maximum threshold for Hue
- $Smin$: Predefined minimum threshold for Saturation
- $Smax$: Predefined maximum threshold for Saturation
- $Vmin$: Predefined minimum threshold for Value
- $Vmax$: Predefined maximum threshold for Value

If this condition is satisfied, then the pixel of the corresponding layer will be 1, otherwise 0. In this case, we can choose any color to be retained through the thresholding process.

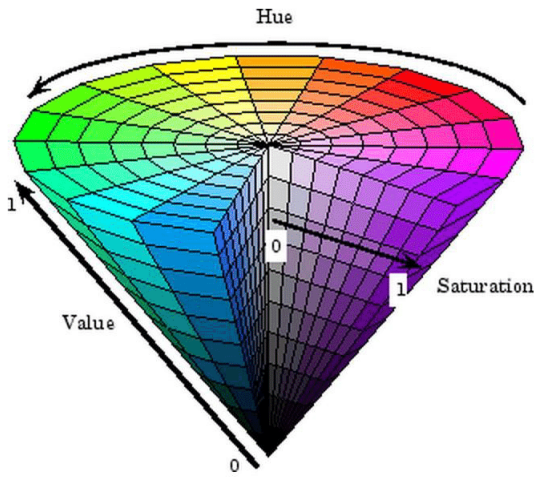


Figure 2: Graph HSV

3.2 Scale reduction

There are various ways to scale down an image. The initial method used during the project was Laplacian Pyramid. However, this method introduces artifacts around the reduced image. Consequently, we adopted a bilinear interpolation method for scale reduction. This technique calculates the pixel value based on the surrounding 4 pixels [5]. Therefore, we can reduce an image from size MxN to size mxn, thereby decreasing the number of pixels by:

$$M * N - (m * n)$$

Nevertheless, we do lose precision by reducing the size of our image. Our machine learning model must find the position of the goal in the image, and if the image is smaller, its prediction will be less accurate. However, in the context of the NAO's control system with the goal frame, we do not require high precision.

On the NAO robot, we have two cameras with different resolutions. The scale reduction enables the model to have uniform dimensions despite the variations between the two cameras.

4 Result

We assume that our input data is an image of dimensions MxN. This RGB image is initially defined on $M * N * 16777216$ different values. Our data preprocessing consists of three steps. The first step is the thresholding of the desired color. Here, we aim for a slightly orangish red from the goal structure. We will take the following threshold values: $Hmin = 0$, $Hmax = 10$, $Smin = 150$, $Smax = 255$, $Vmin = 150$, $Vmax = 255$

Figure 3: goalpost before and after the red thresholding

This initial step already reduces our variable count to $M * N * 2$ with each pixel taking values of either 0 or 255. Additionally, our machine learning model does not have to handle background images, which helps reduce its complexity. The second step after the color thresholding is scaling down from an MxN pixel image to a square 100x100 pixel image. By reducing the scale through a bilinear interpolation method, we reintroduced intermediate values between 0 and 255. At this stage, our variables can take on $100 * 100 * 255 = 2,550,000$ different values.

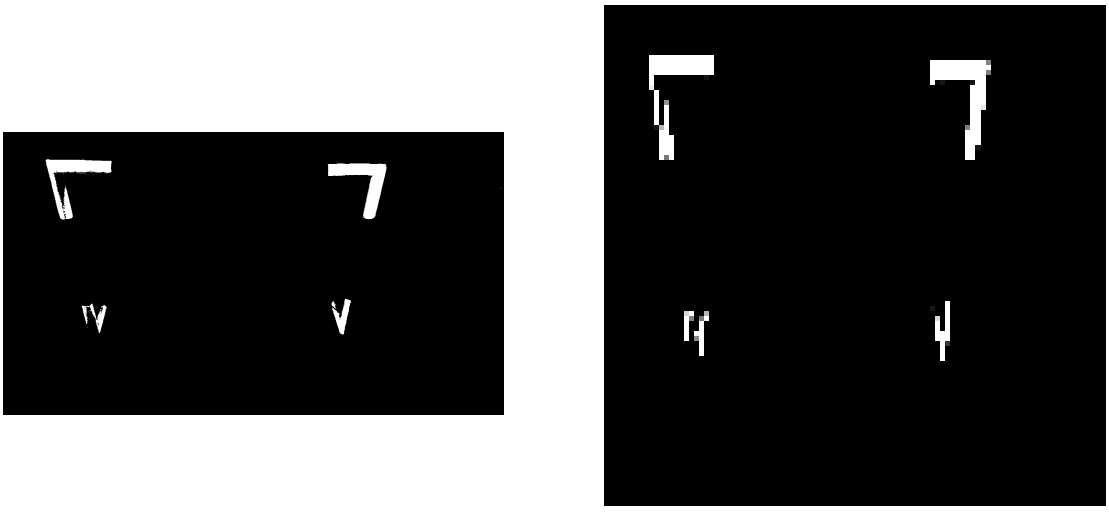


Figure 4: goalpost before and after the resizing

To remove undesired intermediate values and convert back to a binary image, we apply a standard threshold. This reduces our variables to $100 * 100 * 2 = 20,000$ distinct values for our machine learning model.

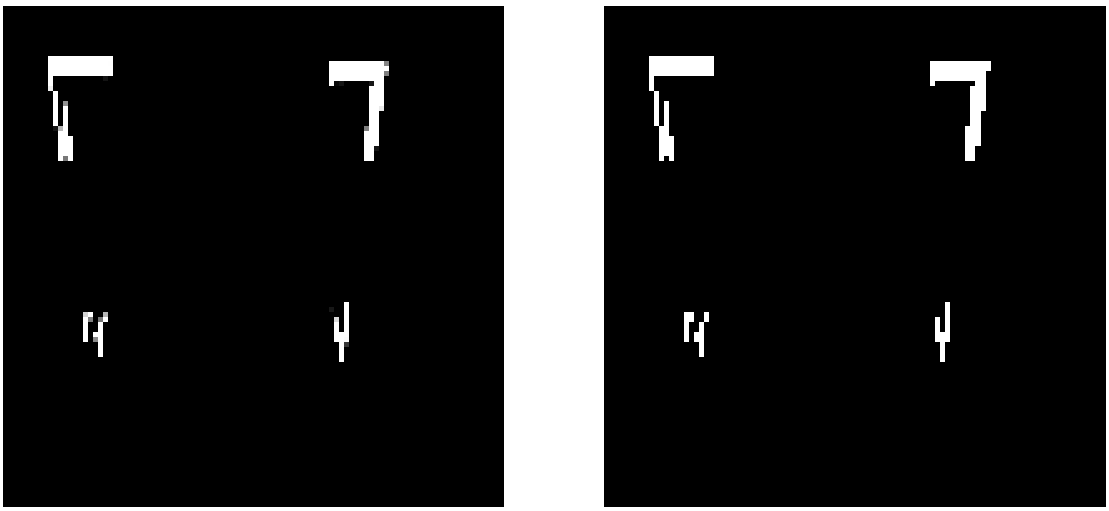


Figure 5: goalpost before and after the thresholding

By proceeding in this way, assuming we have an input image of average quality at 518x518 resolution, our data preprocessing allows for a reduction by $225,086,485$ in possible variables. This transforms the input vector of our model from a shape of $518 * 518 * 3$, with each value in the range of 0 to 255, to an input vector with a shape of $100 * 100 * 1$, where each value is either 0 or 255.

5 Conclusion

Our goal was to minimize the input values of our ML model to have the least complex model possible. One way to reduce input values is through preprocessing. As our ML model is designed to take images as input, we can apply image processing methods to these images. The strategy involves two main parts: thresholding for a desired color and reducing the size of the image. Thresholding the desired color helps reduce the input vector of our model and improves the quality of the input value by separating the goal frame from the rest of the image. Scaling down the image size also contributes to reducing the input vector.

References

[1] T. Mitchell, *Machine Learning*, Science/Engineering/Math, 1997.

[2] S. Zhang, C. Zhang, Q. Yang, *Data preparation for data mining*, Appl. Artif. Intell., p375–381, 2003

[3] S. Krig, *Image Pre-Processing*, Computer Vision Metrics. Apress, Berkeley, CA, 2014.

[4] W. Burger, Mark J. Burge, *Principles of digital image processing : core algorithms*, Springer, 2009.

[5] Rafael C. Gonzalez, Richard E. Woods, Pearson Prentice Hall,*Digital Image Processing*, « Image sampling and Quantization », 2008

Comparative Analysis of Visual Odometry and Lidar Odometry for Robot localization

PITAUD Mathieu
mathieu.pitaud@ensta-bretagne.org
ENSTA Bretagne - Robotique Autonome

***Abstract*—This article aims to present and compare two odometry methods : Visual Odometry and Lidar Odometry. The first part will be devoted to presenting these methods and their variations. Then, I will compare their performance using the Kitti Odometry dataset. The visual Odometry algorithm implemented is based on stereo-vision system and features matching. Lidar odometry is based on Iterative Closest Point algorithm (ICP). We will see that the results are quite good at the beginning but tend to diverge as the mission progresses.**

***Index Terms*—Odometry, Lidar, Camera, Stereo-vision, Localization, Kitti Dataset, Matching, Point-clouds, Iterative Closest Point.**

I. Introduction

In the field of robotics, odometry refers to the estimation of a moving robot's speed and position, which enables the localization of the robot in its environment.

A lot of methods exists to estimate the speed of a robot depending on the sensors. In this paper, we will study two different ways : Visual odometry and Lidar odometry.

Visual odometry relies on images data from one or various cameras (in the case of stereo-vision), and can use matching points or optical flow to estimate motion. Lidar based odometry utilizes laser beams to measure distances and create a 3D points cloud around the robot. By using the points cloud from one instant to the next we can deduce robot's movement.

First, I will detail how the two methods work and their variations. Then, we will compare their performances, using sensors data and true positions from a public dataset.

II. Visual Odometry

Visual odometry aims to estimate the position, speed, and orientation of a moving robot by analyzing sequences of images. It is based on extracting features, from key-points or corners for example, from a frame. And tracking the features over time to know the 2D motion in the image reference frame. Then, triangulating points to get the 3D motion of some key-points. If we suppose that those points are motionless then we can deduce our motion. This is the main idea of visual odometry but it exists a lot of different methods, depending of the type of camera used, for example.

As explained in [1] [2], there are two types of visual odometry algorithms which are feature based VO and appearance based VO.

A. Feature based

As we saw in the previous paragraph. Feature based VO algorithms will search key-points in a frame. Those points are chosen according to several criteria. It has to be distinguishable from other points, which means to have high gradients and contrasts around this point. That is why the corner of an object can be a good candidate, because there are different colors in all the directions for example. A key-point has to be located precisely, that is why an edge is less good than a corner. Some algorithms exists to compute key-points in an image, such as SIFT, ORB, Harris.

The idea is then to find the same key-points in the next image, so we can deduce the motion in 2D in the image reference frame. We use matching algorithms, that compare each features in both image to recognize the same points. We search for the nearest neighbor of each feature, then test several match combinations and keep the one that minimizes errors (in the sense of Euclidean distance).

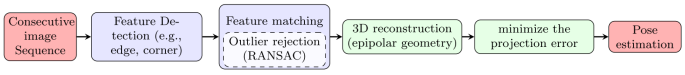


Fig. 1. Illustration of the feature based VO principle [2]

B. Appearance based

Appearance based VO algorithms will look out for intensity of pixels, and the motion of the global intensity in the image to deduce the 2D motion of the scene. It uses optical flow algorithms such as Lucas-Kanade algorithm, that try to compute the motion of brightness/intensity patterns from one image to the next. [1]

Lucas-Kanade algorithm make the assumption that the motion is small between two frames, so we need a high rate camera to use it. Also, the optical flow can be computed on each point (Dense Optical Flow) or just on a few interesting points (Sparse Optical Flow).

C. Stereo Odometry [3] [4]

Stereo-vision odometry uses two cameras (at least) whose relative position to each other is known. So at each instant we have multiples frames of same scene but from slightly different points of view.

We can match key-points in both images and using their intrinsic matrix, we can triangulate to have the key-points 3D coordinates. We can also compute the disparity map using the stereo-vision system, which correspond to the movement of each pixel between the two images. If a pixel has moved a lot then it corresponds to a close object, and if it has moved a little then it corresponds to an object far away. Using the intrinsic parameters we can deduce the depth from the disparity :

Let's notate d the distance between the two cameras, f the focal length given by the intrinsic matrix, and $disp$ the disparity, then for each pixel :

$$depth = \frac{d * f}{disp} \tag{1}$$

To summarize, the feature matching or the optical flow between two images at time t and time t+1 gives us the 2D motion of a point. And by triangulation or disparity map using the stereo-vision system we can change to the 3D coordinates, so we have the 3D motion.

D. Monocular Odometry

Monocular odometry uses only one camera, the disadvantage of this method is the depth issue, as we can not triangulate. Two consecutive images give us the 2D motion, but we can not have the 3D motion. We often need to rely on other sensors such as Lidar, or to use the knowledge of the environment, such as the planarity of the road surface. [2]

III. Lidar Odometry

Lidar based odometry aims to estimate the position, speed, and orientation of a moving robot by analyzing sequences of 3D point clouds. As visual odometry, it is also based on finding the transformation between two point-clouds. Contrary to visual odometry, with Lidar we have directly the 3D coordinates in the Lidar reference frame.

One of the difficulties with Lidar odometry is the Lidar takes a long time to rotate completely. As the robot moves while the Lidar is getting data, it create distorsions in the point-cloud, which we have to take into account to be as accurate as possible.

Different methods are used to align point-clouds such as Iterative Closest Point or the solution used for Lidar Odometry and Mapping based on features. [5] [2]

A. Iterative Closest Point (ICP)

The ICP method aims to find the transformation between to point-clouds. It will compute the transformation that minimizes the distance (the sum of the squares of the distances between the corresponding points) between the two point-clouds. The main advantage of this method is that it is easier to use, as there is no need to extract features.

ICP needs a high computing power as it computes distance for each points. So, in some cases, with a high resolution Lidar, this algorithm may be inefficient as it will be too slow to calculate in real time. Also, we often need to preprocess the data to avoid some noise and outliers that occurs frequently. [5] [2]

B. Lidar Odometry and Mapping (LOAM)

LOAM aims to estimate the position of the robot but also to create a map at the same time. The first part of the algorithm is based on finding features in the point-cloud such as edges and corners. And try to recognize those features in the next point-cloud.

First, we need to extract features, in both papers [6] [5], they use the smoothness of the local surface as a feature. The smoothness of a point depends of the distance to the other nearby points, often the five nearest in each direction. Smoothness is calculated by summing the distance to each nearby point. As a result, points located at a corner will have a high smoothness, and points located in a flat surface will have a very low smoothness.

So, we have two groups of features, points that are more likely edges or corners and points that are more likely on wall, or other flat surfaces. [5] [6]

The second step is to find the correspondences between features at time t, and features at time t+1. In the paper [6], they superpose the two point-clouds and try to find edges lines or planar areas. They keep the solution that minimizes errors.

IV. Dataset

In order to test and verify previously detailed odometry methods we need data from a real environment. We will use the Odometry Benchmark of the Kitti Vision Benchmark Suite dataset, downloadable here [7], which comes from a collaboration between Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago [8].

This dataset provides sensors data from a moving car in various environments. The vehicle is equipped with 3D Lidar, GPS, IMU and a stereo-vision system composed of four cameras (2 colors, and 2 monochromes), as we can see on Figure 2.

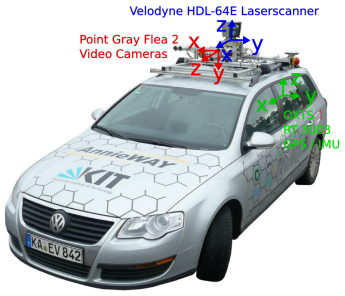


Fig. 2. Illustration of the Kitti car's sensors [7]

The main advantages of this dataset are the calibration data provided, data from cameras and Lidar corresponding to the same mission, and the large number of sequences, some with the true position given for comparison purposes and performance characterization.

V. Results

A. Visual Odometry

To write the code I took the example of a github project [9]. We will use a feature based method using stereo-vision system.

First, we can find the matching points between a image and the next one. The comparison of the matching points coordinates between the two images give us the 2D motion in the camera reference frame. An example of results is shown on Figure 3.

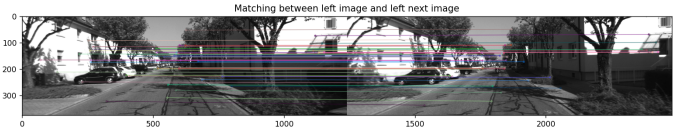


Fig. 3. Matching points

In order to have the 3D coordinates, we compute the disparity map between the left and right images. We can deduce the depth map using the equation 1 established previously. See Figures 4 and 5.

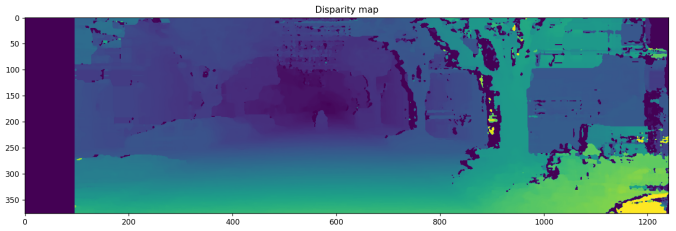


Fig. 4. Disparity map

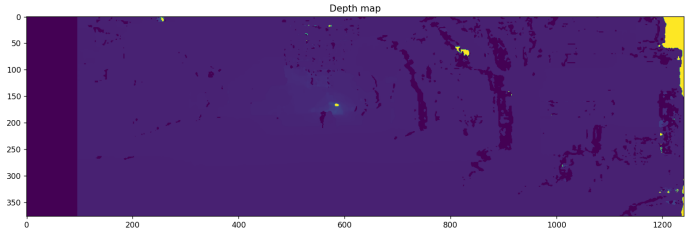


Fig. 5. Depth map

We are able to determine the 3D coordinates using the following equations that comes from the intrinsic matrix definition $p = MP$:

$$X = \frac{(u - u_0) * Z}{f_x}; Y = \frac{(v - v_0) * Z}{f_y} \tag{2}$$

Where (u, v) are the 2D coordinates of the considered point (in pixels), (u_0, v_0) the center of the image (in pixels), and (f_x, f_y) the focal length divided by the length of a pixel respectively according the x-axis ans the y-axis.

Then we use the $cv2.solvePnPRansac()$ function that gives us our motion (translation and rotation matrix) in the camera reference frame. The inputs of this function are the 3D coordinates in the first image and the 2D coordinates of the corresponding points in the second image.

We transform the coordinates in the world reference frame to compare with the ground truth. The results are shown in Figure 7.

B. Lidar Odometry

We will use the Iterative Closest Point method. To avoid expensive calculations we will sub-sample the point-cloud (initially composed of 100 000 points), to take only 15 000 points chosen randomly. Otherwise the computation would take to much time.

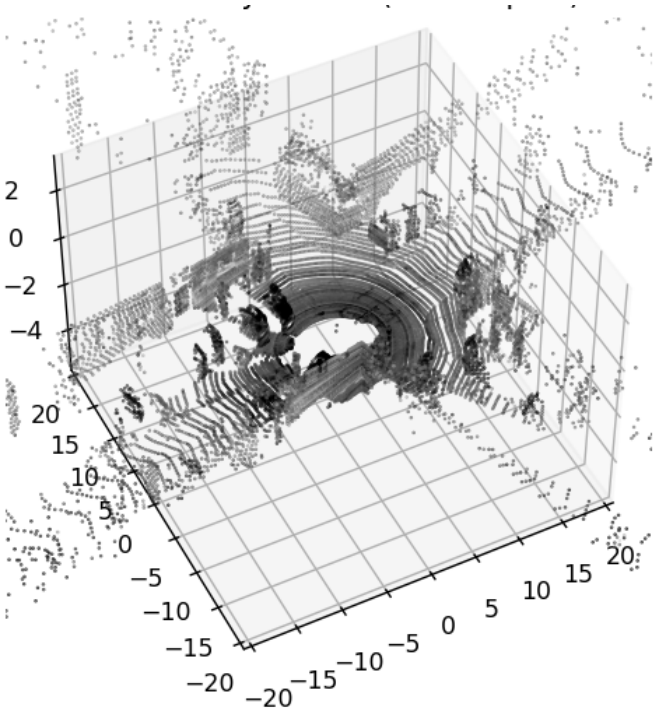


Fig. 6. Example of a point-cloud given by the Velodyne Scan (20 000 points)

On Figure 6 there is an example of the point-cloud given by the Velodyne Scan. Here the car is located at a crossroad.

Then, I use the $pipelines.registration.registration\_icp$ function of the open3d python library, that computes the transformation matrix using ICP method, where the inputs are only two consecutive point-clouds.

We transform the coordinates in the world reference frame to compare with the ground truth. The results are shown in Figure 7.

C. Comparison

In the Figure 7 that presents the results. We can see that both methods manage to correctly follow the ground-truth trajectory at the beginning. But as the simulation progresses, we observe that the estimated trajectories slowly deviate from the truth.

The algorithms always detect turns and straight lines, and the final shape of the trajectory looks like the ground truth.

Trajectory errors come from small accumulated errors in the rotation matrix. If there is a tiny error in the rotation matrix, even if we detect that we are moving in a straight line, the trajectory will deviate from the truth. With this method, the trajectory will always end up diverging. To solve this problem, we would have to use other sensors or close the loop (detect that we went back to the same place).

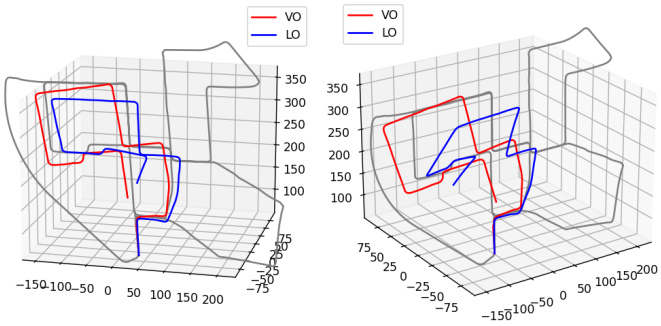


Fig. 7. Ground-truth and estimated trajectories (Kitti odometry dataset 0)

To improve the results, we could have taken into account the fact that the car is moving on a road, so its rotation vector is constrained. We could easily consider that the car will not have pitch or roll.

We can also note that using more advanced algorithms and optimizations would probably have produced better results. A possible improvement for the Lidar would be to have a preprocess algorithm to remove outliers, and to use a feature-based algorithm instead of the Iterative Closest Point which is quite basic.

To conclude, in this case both solutions gave good results. However, depending on the environment and the mission one solution could be better than the other. High accuracy Lidar are often more expensive than cameras. But Lidar are more robust whereas visual odometry can be affected by the rain, or by poor lightning conditions.

REFERENCES

[1] Mohammad OA Aqel, Mohammad H Marhaban, M Iqbal Saripan, and Napsiah Bt Ismail. Review of visual odometry: types, approaches, challenges, and applications. *SpringerPlus*, 5:1–26, 2016.

[2] Sherif A. S. Mohamed, Mohammad-Hashem Haghbayan, Tomi Westerlund, Jukka Heikkonen, Hannu Tenhunen, and Juha Plosila. A survey on odometry for autonomous navigation systems. *IEEE Access*, 7:97466–97486, 2019. https://doi.org/10.1109/ACCESS.2019.2929133 doi:10.1109/ACCESS.2019.2929133.

[3] Davide Scaramuzza and Friedrich Fraundorfer. Visual odometry [tutorial]. *IEEE Robotics Automation Magazine*, 18(4):80–92, 2011. https://doi.org/10.1109/MRA.2011.943233 doi:10.1109/MRA.2011.943233.

[4] D. Nister, O. Naroditsky, and J. Bergen. Visual odometry. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I, 2004. https://doi.org/10.1109/CVPR.2004.1315094 doi:10.1109/CVPR.2004.1315094.

[5] Han Wang, Chen Wang, Chun-Lin Chen, and Lihua Xie. F-loam : Fast lidar odometry and mapping. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4390–4396, 2021. https://doi.org/10.1109/IROS51168.2021.9636655 doi:10.1109/IROS51168.2021.9636655.

[6] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and systems*, volume 2, pages 1–9. Berkeley, CA, 2014.

[7] Kitti vision benchmark suite. URL: https://www.cvlibs.net/datasets/kitti/eval_odometry.php.

[8] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[9] Github project on visual odometry. URL: https://github.com/NafBZ/Stereo-Visual-Odometry/tree/master.

Water-surface obstacle detection using machine learning

RAVAIN Louis
LABSTIC
ENSTA Bretagne
Brest, FRANCE
louis.ravain@ensta-bretagne.org

***Abstract*—Water Surface object detection is very significant in the advancemement of autonomous navigation for ships and boats. It's the maritime equivalent of detecting cars, traffic signs, pedestrians, etc., for the development of autonomous vehicles. However, unlike the automotive sector, sourcing an appropriate dataset for water surface object detection poses a challenge. Therefore, in this paper, we propose a dataset composed of 8325 images, representing 5 differents common objects found on the water surface, under many different conditions. We will also use this dataset to train a machine learning model, YOLOv8 and test the resulting model on the waters of Lake Guerlédan.**

***Index Terms*—Machine learning, dataset, object detection, YOLO**

I. Introduction

In recent years, autonomous driving has seen major progress, and autonomous water navigation is a topic of interest. Indeed, water transportation is far better for the environmment, as it is less CO2 intensive than air or even road transportation. At the same time, the advancements in artificial intelligence allows us to imagine more efficient ships. As an important component to autonomous navigation, water-surface object detection is key to the development of these autonomous ships. Indeed, if we want to preserve the safetyness of water transportation, we have to be able to detect any obstacle that might be on the ship's path. With the continuous development of deep learning technology, object detection algorithms based on convolutional neural networks (CNN) have been proposed one after another. CNN, a type of deep feedforward neural network, has found numerous successful applications in tasks such as image classification, object detection, and object segmentation across various image and video domains. Water surface object detection technology is increasingly adopting this approach due to its high accuracy, fast speed, and strong generalization capabilities. Detecting objects on the water surface requires high real-time performance and recognition accuracy, especially for objects located at long distances. Despite the actual size of distant water surface objects potentially spanning tens or even hundreds of meters, they may only occupy a few pixels on the imaging plane. Among the various object detection algorithms currently available, the Single-shot Multibox Detector (SSD) algorithm stands out for its speed and accuracy. However, its performance in detecting small objects is not as satisfactory as desired, particularly when applied to water surface object detection at long distances. Currently, object detection primarily relies on traditional vision methods, as they offer predictable results and a fast detection. However, they often suffer from various limitations such as restricted field of view, presence of blind spots, and inability to gather global information. Therefore, in this work, we will propose a new dataset and new associated lightweight machine learning model based on the YOLOv8 architecture to combine the advantages of both approaches. The key feature of YOLO [3] is its single-stage detection approach, which is designed to detect objects in real time and with high accuracy. Unlike two-stage detection models, such as R-CNN [1], that first propose regions of interest and then classify these regions, YOLO processes the entire image in a single pass, making it faster and more efficient.

The main contributions of this paper are as follows :

- Build a coherent and balanced dataset
- Propose a lightweight trained model based on YOLOv8 COCO-trained model
- Benchmark this new model on the water of Lake Guerlédan

Therefore, the remainder of this paper is organized as follow : Section II describes previous work in water surface object detection and dataset creation. Section III presents the dataset created and how it was built. Section IV introduces the trained model and evaluation indicators, before analysing the experimental results. Finally, section V presents the conclusions of this paper and expands on future possible works.

II. State of the art

A. Existing datasets

Generic image-based datasets, such as MS COCO [5], ImageNet, and Places2, can be considered for water surface detection tasks. However, their utility in this domain is limited due to the scarcity of relevant categories and images. For example, MS COCO [5] includes only one category (boat) related to water surface detection, with 3,146 images, which may not be sufficient for effective neural network training. Similarly, ImageNet provides annotations for only four

Identify applicable funding agency here. If none, delete this.

categories (catamaran, trimaran, container-ship, and aircraft-carrier) related to water surface objects, and these images may not accurately represent real-world water surface conditions. Additionally, Places2 contains just five categories (harbor, lake, loading-dock, water, and river) relevant to water surface detection. Overall, the lack of diverse water surface obstacles in these datasets limits their applicability for water surface object detection tasks. On the other hand, at present, there are only a few datasets available for detecting objects on the water surface. The Boat-types-recognition dataset is currently the only publicly available image-based dataset in this domain. It comprises 1,462 images of water surfaces, featuring three categories of common objects: boats (including gondolas, inflatable boats, kayaks, paper boats, and sailboats), ships (such as cruise ships, ferry boats, and freight boats), and buoys. While this dataset offers ample variation in water environments and shooting times, it lacks annotations for object detection. A recent paper [6] proposed a larger and better dataset, called Water Surface Object Detection Dataset. While it is a step in the right direction, the performances of the models trained on this dataset were found to be lackluster in our testing environment. Therefore, there was a need for (yet) another water surface object detection dataset

B. Methods

It is widely acknowledged that early generic object detection methods faced challenges in feature extraction from images, resulting in subpar precision and speed of object detection. However, since 2012, the advent of deep learning has led to the emergence of highly efficient CNN-based detectors, broadly categorized into two groups: two-stage and one-stage object detection methods. The well-known two-stage approaches include the R-CNN series [1], exemplified by Faster R-CNN [2], Mask R-CNN, and Cascade R-CNN. On the other hand, prominent one-stage methods include Yolo [3] and SSD [10]. Furthermore, one-stage detectors can be adapted into anchor-free variants, such as CenterNet or newer versions of YOLO.

The advent of deep learning has significantly propelled advancements in this field. Due to the inherent challenges posed by variations in size, appearance, and disturbances, unsupervised methods have been found to be severely limited, leading to a prevalent use of supervised techniques. Reference [11] introduced an architecture employing Fast R-CNN for ship identification and classification. Additionally, [12] presented a hybrid ship detection method integrating deep learning approaches, utilizing Deep Neural Networks (DNNs) and Region Proposal Networks (RPNs) to delineate 2D bounding boxes of target ships. Reference [13] devised a rapid detection method for surface objects based on ResNet, achieving object detection speeds of up to 32.4 frames per second (FPS). Reference [14] utilized FCN for surface obstacle detection, showcasing robustness in performance. Reference [15] (2019) proposed an enhanced RBox-based framework for water surface target detection, optimizing detection accuracy, recall rate, and precision. Reference [16] introduced a ship detection algorithm utilizing an improved YOLO and multi-feature approach, incorporating SIFT feature reduction via multi-dimensional scaling (MDS) and optimizing feature matching through RANSAC. Reference [17] proposed a real-time water surface object detection method based on an enhanced Faster R-CNN, integrating low-level features with high-level features to enhance detection accuracy. Reference [18] employed deep residual networks and cross-layer jump connections to extract advanced ship features, enhancing object recognition performance. In 2020, [19] proposed a method based on YOLOv2 for detecting small ships, applicable for identifying various obstacles on the water surface. Reference [20] introduced H-YOLO for ship detection based on a preselected region of interest network, leveraging hue, saturation, and value (HSV) differences to distinguish suspected ship areas from the background. Reference [21] proposed YOLOv3-2SMA for real-time and high-precision object detection in dynamic aquatic environments. Reference [22] (2021) enhanced YOLOv3 for ship detection in inland waterways, achieving improvements in mean average precision (mAP) and frames per second (FPS). Recently, [23] introduced ShipYolo to address missed inspections of small-scale ships, employing a novel amplified receptive field module with dilated convolution and Max-Pooling for improved spatial information acquisition and target space displacement robustness. However, many of the aforementioned methods, primarily designed for static cameras in port management, may not be suitable for application in autonomous ships equipped with shipborne surveillance systems [22], and despite advancements, efficiency and accuracy remain persistent challenges.

III. Creation of the dataset

As we have seen before, we needed a more suitable dataset for our purposes. This section will discuss the creation of this dataset.

A. Presentation of the dataset

The dataset comprises 8325 images, and 5 classes : Boat, Ship, Buoy, Ball and Breakwater. Images were taken from several different open-sourced dataset, including the WSOD dataset shared by [6]

As we can see in figure 4, the dataset is fairly balanced, which will reduce our model's ability to misidentify objects.

B. Choices made

We have chosen to build a rather limited but balanced dataset. Indeed, we only have 4 classes, but each classes is well represented, with only the Breakwaters being under represented a bit, which highlights an issue future works could address.

IV. Testing the model and the dataset

We will now test our dataset and see if the performances of our model meet our requirements. We have chosen the YOLOv8 algorithm fo its speed, its efficiency even on low-powered devices such as the Raspberry Pi 4, and its ease of use.

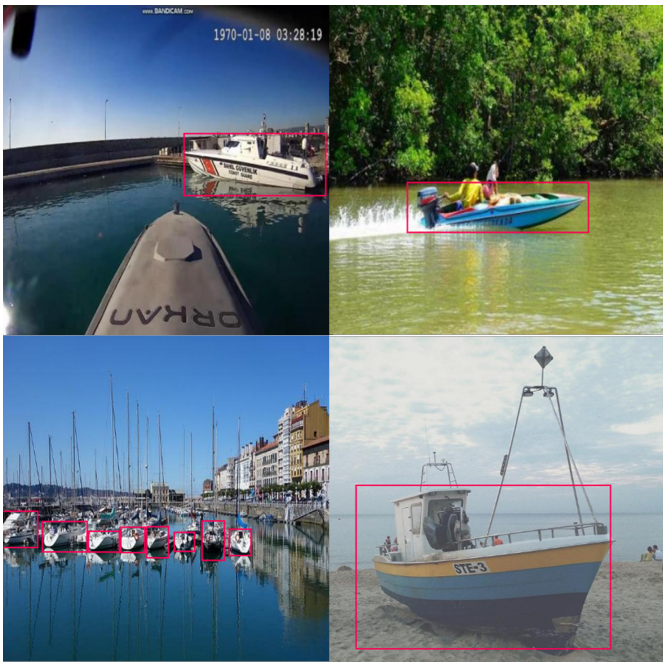


Fig. 1. Example of boat images

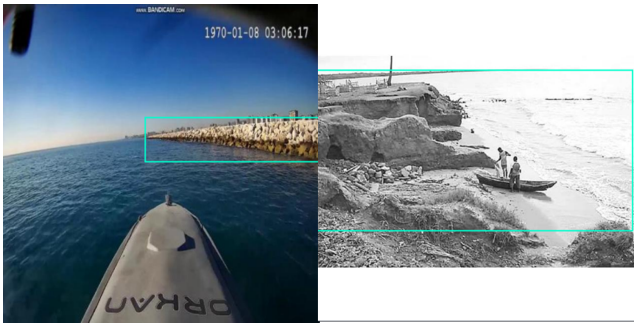


Fig. 2. Example of waterbreaks images

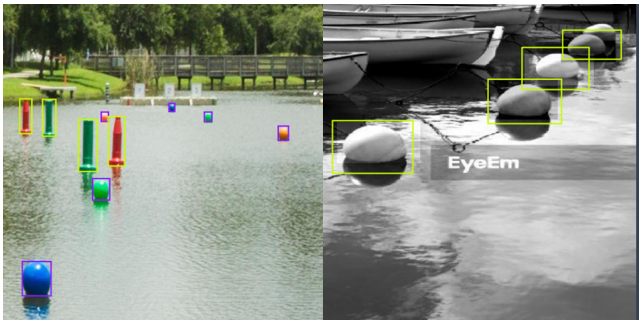


Fig. 3. Example of buoys images

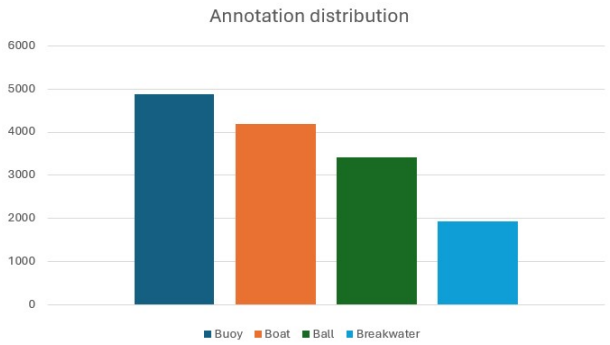


Fig. 4. Annotation Distribution

A. *Presentation of YOLOv8*

The You Only Look Once (YOLO) [3] algorithm, introduced in 2015 by Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi, represents a major advancement in real-time object detection. Unlike its predecessor, the Region-based Convolutional Neural Network (R-CNN) [1], YOLO operates in a single pass, directly classifying objects by employing a single neural network to predict bounding boxes and class probabilities using the entire image as input.

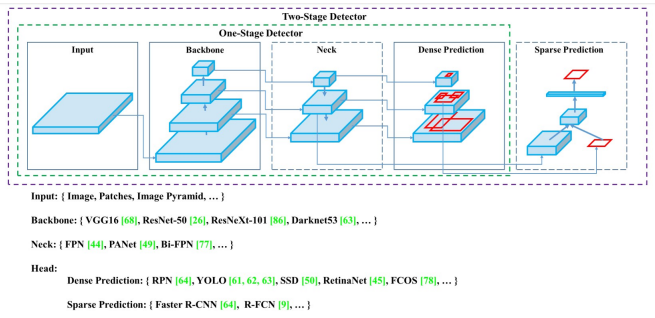


Fig. 5. General Object detector architecture [9]

However, YOLO v1 suffered from localization errors and low recall for small objects.

In response to these limitations, YOLO v2 was introduced in 2016. This version addressed some of the shortcomings of its predecessor by implementing anchor boxes to handle object scale variations, incorporating batch normalization and high-resolution classifiers, and supporting multi-scale training for improved accuracy on small objects.

Subsequent iterations further improved upon the YOLO architecture. YOLO v3, introduced in 2018, adopted a variant called Darknet-53 as the backbone network and introduced feature pyramid networks (FPN) for detecting objects at different scales. YOLO v4 (2020) introduced significant architectural changes, including the CSPDarknet53 backbone and PANet feature fusion, leading to state-of-the-art performance on various object detection benchmarks.

In parallel, YOLO v5 (2020), developed by Ultralytics, introduced a different architecture based on the CSPNet and

focused on model scaling and efficiency. It achieved competitive performance with faster training times compared to previous versions.

YOLOv6 is another unofficial version of the YOLO series introduced in 2022 by Meituan – a Chinese shopping platform. The company targeted the model for industrial applications with better performance than its predecessor. The significant differences include anchor-free detection and a decoupled head, which means one head performs classification. In contrast, the other conducts regression to predict bounding box coordinates. The changes resulted in YOLOv6(nano) achieving an mAP of 37.5 at 1187 FPS on the COCO dataset and YOLOv6(small) achieving 45 mAP at 484 FPS.

In July 2022, a group of researchers released the open source model YOLOv7, the fastest and the most accurate object detector with an mAP of 56.8% at FPS ranging from 5 to 160. Extended Efficient Layer Aggregation Network (E-ELAN) forms the backbone of YOLOv7, which improves training by letting the model learn diverse features with efficient computation. Also, the model uses compound scaling for concatenation-based models to address the need for different inference speeds.

And in january 2023, YOLOv8 was released, which boasts higher accuracy and faster speed. However, without any published papers, there is no concrete evidence, despite experimental indications suggesting its validity.

Our model is based on the pre-trained yolov8n.pt model, which is the smallest model proposed by Ultralytics. Indeed, our aim was to develop a lightweight model capable of running inferences within a reasonable time frame on a Raspberry Pi 4.

We can see a comparison of the differents models sizes and speed in fig 6 below.

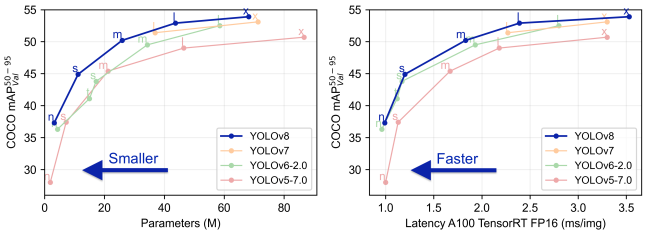


Fig. 6. Comparison of yolo models [3]

B. Training results and test on our benchmark

As we can see in the figure 9 above , we have very promising results when detecting boats and buoys. Nevertheless, the absence of dock detection highlights a notable area for enhancing our dataset. However, we also have some errors in difficult conditions, as shown by the figure 10 below . Indeed, when the objects are too small (or too far), our model doesn't detect them. On the other hand, when the light is such that there are a lot of reflections, our model makes wrong predictions.

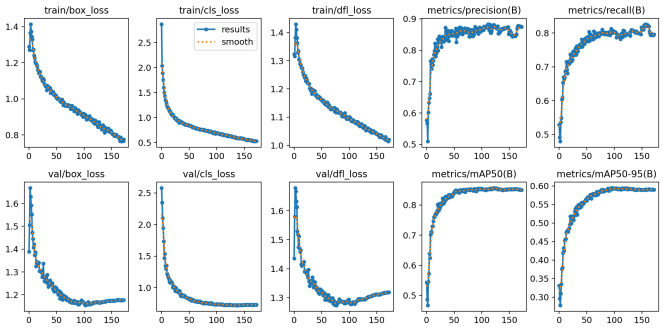


Fig. 7. Metrics of our testing process

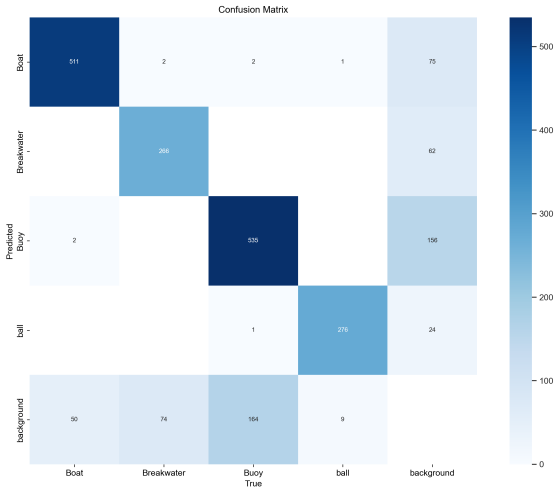


Fig. 8. Confusion Matrix

Overall, the results are good, with mistakes being relatively rare on our video tests, and they happen in difficult conditions, which is not surprising. On top of that, it takes about 500ms to run an inference on a Pi 4, and about 20ms on a more powerful laptop equipped with an rtx2080, which is very reasonable.

V. Conclusion

The goal of this paper is to improve the fiability of water surface object detection. First a new dataset was created,

Fig. 9. Results achieved on our testing videos

Fig. 10. No detection and Misclassification on our testing videos

better suited to our needs. Then, a new model based on the YOLOv8 algorithm was trained and tested on the waters of Lake Guerledan, where it showed superior results to the base model trained on the COCO dataset. However, future works should focus on enhancing the dataset further : adding more classes, and creating a benchmarking dataset for comparing future models among themselves. Indeed, object detection and machine learning is an ever-evolving research topic, witnessing new and improved models.

REFERENCES

[1] Ross Girshick, Jeff Donahue, Trevor Darrell and Jitendra Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation", arXiv 2014.

[2] Ross Girshick, "Fast R-CNN", arXiv, 2015

[3] Joseph Redmon, Santosh Divvala, Ross Girshick and Ali Farhadi, "You Only Look Once: Unified, Real-Time Object Detection", arXiv 2016

[4] Li, A., Zhu, X., He, S. et al. "Water surface object detection using panoramic vision based on improved single-shot multibox detector." EURASIP J. Adv. Signal Process. 2021

[5] Tsung-Yi Lin and Michael Maire and Serge Belongie and Lubomir Bourdev and Ross Girshick and James Hays and Pietro Perona and Deva Ramanan and C. Lawrence Zitnick and Piotr Dollár, "Microsoft COCO: Common Objects in Context", arXiv 2015

[6] Zhou Z, Sun J, Yu J, Liu K, Duan J, Chen L, Chen CLP. "An Image-Based Benchmark Dataset and a Novel Object Detector for Water Surface Object Detection." Front Neurorobot. 2021 Sep 24. doi: 10.3389/fnbot.2021.723336.

[7] Buoy Dataset, Open Source Dataset, https://universe.roboflow.com/buoy/buoy-jv26h, Oct 2022

[8] Wang, Chien-Yao and Liao, Hong-Yuan Mark, "YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information" arXiv preprint arXiv:2402.13616, 2024

[9] Alexey Bochkovskiy and Chien-Yao Wang and Hong-Yuan Mark Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection", arXiv, 2020

[10] Liu, Wei and Anguelov, Dragomir and Erhan, Dumitru and Szegedy, Christian and Reed, Scott and Fu, Cheng-Yang and Berg, Alexander C., "SSD: Single Shot MultiBox Detector", Lecture Notes in Computer Science (21-37), 2016, doi:10.1007/978-3-319-46448-0_2

[11] Yang, J., Xiao, Y., and Fang, N. (2017). "An object detection and tracking system for unmanned surface vehicles," in Procceedings of Target and Background Signatures (Warsaw), 214–220.

[12] Yao, Y., Jiang, Z., Zhang, H., Zhao, D., and Cai, B. (2017). Ship detection in optical remote sensing images based on deep convolutional neural networks. J. Appl. Remote Sens. 11, 042611. doi: 10.1117/1.JRS.11.042611

[13] Chae, K. H., Moon, Y. S., and Ko, N. (2017). "Visual tracking of objects for unmanned surface vehicle navigation," in International Conference on Control, Automation and Systems (Jeju), 335–337. doi: 10.1109/ICCAS.2016.7832338

[14] Qin, Y., and Zhang, X. (2018). Robust obstacle detection for unmanned surface vehicles. Proc. SPIE 10611, 2199–2207. doi: 10.1117/12.2285607

[15] An, Q., Pan, Z., Liu, L., and You, H. (2019). DRBox-v2: an improved detector with rotatable boxes for target detection in SAR images. IEEE Geosci. Remote Sens. 57, 8333–8349. doi: 10.1109/TGRS.2019.2920534

[16] Sr, Y. Z. Sr, J. S., Sr, L. H., Sr, Q. Z., and Sr, Z. D. (2019). "A ship target tracking algorithm based on deep learning and multiple features," in Proceedings of the Twelfth International Conference on Machine Vision (Amsterdam), 1143304.

[17] Zhang, L., Zhang, Y., Zhang, Z., Shen, J., and Wang, H. (2019). Real-time water surface object detection based on improved faster R-CNN. Sensors 19, 3523. doi: 10.3390/s19163523

[18] Liu, B., Wang, S. Z., Xie, Z. X., Zhao, J. S., and Li, M. F. (2019). Ship recognition and tracking system for intelligent ship based on deep learning framework. Int. J. Mar. Navig. Saf. Sea Transport. 13, 699–705. doi: 10.12716/1001.13.04.01

[19] Chen, Z., Chen, D., Zhang, Y., Cheng, X., Zhang, M., and Wu, C. (2020b). Deep learning for autonomous ship-oriented small ship detection. Saf. Sci. 130, 104812. doi: 10.1016/j.ssci.2020.104812

[20] Tang, G., Liu, S., Fujino, I., Claramunt, C., Wang, Y., and Men, S. (2020). H-YOLO: a single-shot ship detection approach based on region of interest preselected network. Remote Sens. 12:4192. doi: 10.3390/rs12244192

[21] Li, X., Tian, M., Kong, S., Wu, L., and Yu, J. (2020). A modified YOLOv3 detection method for vision-based water surface garbage capture robot. Int. J. Adv. Robot. Syst. 17, 172988142093271. doi: 10.1177/1729881420932715

[22] Jie, Y., Leonidas, L., Mumtaz, F., and Ali, M. (2021). Ship detection and tracking in inland waterways using improved YOLOv3 and Deep SORT. Symmetry 13, 308–326. doi: 10.3390/sym13020308

[23] Han, X., Zhao, L., Ning, Y., and Hu, J. (2021). ShipYolo: an enhanced model for ship detection. J. Adv. Transport. 2021, 1060182. doi: 10.1155/2021/1060182

Physics-informed neural networks

REN Kevin
FISE 2024 Autonomous Robotics
ENSTA Bretagne

***Abstract*—In recent years, the integration of physics principles with neural networks has garnered significant attention in various scientific and engineering domains. Physics-informed neural networks (PINNs) have emerged as a powerful framework that incorporates prior knowledge of physical laws into the training process of neural networks (NNs). Specifically, they facilitate the consideration of physical phenomena described by ordinary differential equations (ODEs) or partial differential equations (PDEs). While various numerical methods exist for solving these equations, such as finite element or finite difference methods for PDEs, and Runge-Kutta (RK) methods for ODEs, these approaches are often computationally intensive and lack real-time capability.**

PINNs offer a solution to this challenge. Despite being currently less accurate than traditional methods, initial results demonstrate that these NNs can provide reliable results with significantly reduced computation time—up to several hundred times faster than classical numerical methods. The key advantage of PINNs over classical NNs lies in mitigating the stochastic effects inherent in the latter. Classical NNs may yield disparate results across different training runs, even when using the same dataset. Moreover, the utilization of PINNs substantially diminishes the required dataset size, leveraging the inherent knowledge of the physical model being represented.

I. Introduction

Time-domain simulations play a pivotal role in both scientific exploration and engineering design across a spectrum of disciplines. From understanding the behavior of physical systems to predicting complex phenomena, the ability to simulate temporal dynamics accurately is indispensable. Traditionally, numerical methods such as finite element methods (FEM) have served as the cornerstone for time-domain simulations, providing robust solutions to systems governed by ordinary and partial differential equations. However, as computational demands escalate and the need for real-time analysis grows, there arises a pressing need for innovative approaches that can deliver accurate results swiftly.

Physics-informed neural networks have emerged as a promising paradigm at the intersection of physics-based modeling and deep learning. These networks capitalize on the complementary strengths of both fields, harnessing the power of neural networks to incorporate prior knowledge of physical laws into the training process. Unlike traditional numerical methods such as FEM or finite difference methods (FDM), PINNs leverage the representational power of NNs to learn the underlying physics directly from data, bypassing the need for discretization of the entire domain. This integration enables PINNs to tackle time-domain simulations with unprecedented efficiency and flexibility.

Traditionally, power system engineers have relied on various Model Order Reduction (MOR) methodologies [1, 5] and modal analysis techniques to overcome the computational burden and complexity of large models.

While such equivalency tools can produce compact surrogate models, standard MOR has significant drawbacks in certain applications. For example, the majority of MOR techniques used in power systems are only applicable for linear systems. Moreover most MOR methods are projection based, and the resulting dynamical model must still be directly simulated by an ODE solver. In contrast to that, Neural Networks (NNs) have the capacity to universally approximate any continuous function with an arbitrary degree of accuracy [3]. Therefore, they are able to overcome all of the aforementioned drawbacks, and they have become a popular alternative to classical MOR approaches. Recently, the so-called Physics-Informed Neural Network (PINN) was proposed as a framework for directly mapping dynamical system inputs (initial conditions) to system outputs (state trajectories).

In this article, we examine the significance of time-domain simulations in scientific and engineering contexts especially in robotics where state estimation (navigation) is fundamental, with a focus on systems described by ODEs. We discuss the inherent challenges of traditional numerical solvers in capturing the dynamics of ODE-based systems, including their computational overhead and limitations in real-time analysis.

II. Physics-informed neural networks

A. Understanding Neural Netowrks

Neural networks can be mathematically represented as a composition of functions, where each layer performs a linear transformation followed by a nonlinear activation function. Let's consider a simple feedforward neural network with L layers. At each layer l, the input $\boldsymbol{z}^{(l)}$ is transformed using a weight matrix $W^{(l)}$ and a bias vector $\boldsymbol{b}^{(l)}$ followed by an activation function $\sigma^{(l)}$:

$$\boldsymbol{z}^{(l+1)} = \sigma^{(l)}(W^{(l)}\boldsymbol{z}^{(l)} + \boldsymbol{b}^{(l)})$$

$\boldsymbol{z}^{(0)}$ is equal to the input vector $\boldsymbol{x}$ of the NN and $\boldsymbol{z}^{(L)}$ is the output $\boldsymbol{y}$. During training, the objective is to adjust the weights and biases of the NN in order to minimize the loss function L.

B. Loss function for Physics-Informed Neural Networks

Physics-Informed Neural Networks represent a novel framework that integrates the power of NNs with the principles of physics, offering a promising approach for modeling complex

dynamical systems. PINNs leverage the expressive capacity of NNs to learn and approximate the underlying physical laws. This section provides a detailed explanation of PINNs along with the associated mathematical formalism.

Consider a general ordinary differential equation (ODE) of the form:

$$\frac{du}{dt}(t) = F(t, u(t)) \quad (1)$$
$$u(t_0) = u_{t_0} \quad (2)$$

where u represents the unknown function, $t_0 \in \mathbb{R}$ and F is known.

To solve this ODE using PINNs, we employ a NN to approximate the solution u.

Let $u_\theta(t)$ denote this NN parameterized by θ. It aims to predect $u(t)$ given the initial condition at $t = t_0$. The NN architecture, typically involving feedforward or recurrent NNs, is chosen based on the problem at hand. In this work, a dense architecture is employed.

PINNs enforce the ODE constraint by augmenting the loss function with terms that encode the differential equation. This is achieved by taking derivatives of the neural network solution using automatic differentiation (AD) and enforcing the ODE at a set of collocation points, which are chosen to be equal to $(t_i)_i$, the data points. By aligning the collocation points with the data points, the PINN is trained to simultaneously fit the observed data and satisfy the differential equation at the same set of points. The loss function for training the PINN is then defined accordingly:

$$L = \lambda_1 L_{\text{data}} + \lambda_2 L_{\text{ode}} + \lambda_3 L_{\text{ic}} \quad (3)$$

where

$$L_{\text{data}} = \frac{1}{N}\sum_i (u(t_i) - u_\theta(t_i))^2 \quad (4)$$
$$L_{\text{ode}} = \frac{1}{N}\sum_i \left(\frac{du_\theta(t_i)}{dt} - F(t_i, u_\theta(t_i))\right)^2 \quad (5)$$
$$L_{\text{IC}} = (u(t_0) - u_\theta(t_0))^2 \quad (6)$$

where the first term represents the data fidelity term, ensuring that the neural network solution matches the observed data points $(t_i, u_i)_i$, the second term represents the physics-informed loss, enforcing the ODE at the $(t_i)_i$ and the last term represents the error of the initial condition, ensuring that the neural network solution satisfies the specified initial condition. The coefficients $(\lambda_i)_i$ serve to balance the influence of the different loss terms.

In fact, this type of PINNs does not perform very well. The solution u_θ can converges to a solution that verify the ODE but not anymore the initial condition as t tends to infinity. This will be demonstrated later in section IV.B.

A more effective approach involves changing the inputs. Let $u_\theta(\Delta t_i, u_{t_i})$ denote this NN parameterized by θ. It aims to predect $u(t_i + \Delta t_i)$ given the value at $t = t_i$. Therefore, every $u(t_i)$ can be considered as initial condition. By denoting $u_\theta(\Delta t_i, u_{t_i}) = \hat{u}_{t_i+\Delta t_i}$ equations (4)(5)(6) can be rewritten as follow :

$$L_{\text{data}} = \frac{1}{N}\sum_i (u_{t_i+\Delta t_i} - \hat{u}_{t_i+\Delta t_i})^2 \quad (7)$$
$$L_{\text{ode}} = \frac{1}{N}\sum_i \left(\frac{d\,\hat{u}_{t_i+\Delta t_i}}{dt} - F(t_i + \Delta t_i, \hat{u}_{t_i+\Delta t_i})\right)^2 \quad (8)$$
$$L_{\text{IC}} = \frac{1}{N}\sum_i (u_{t_i} - \hat{u}_{t_i})^2 \quad (9)$$

III. Implementation

A. Case study

We demonstrate the proposed PINNs on Harmonic oscillator. The harmonic oscillator serves as a fundamental model in physics, widely utilized to describe various oscillatory phenomena across different disciplines. It described by the following ODE:

$$\frac{d^2x}{dt^2} + \omega^2 x = 0 \quad (10)$$

This ODE is equivalent to

$$\dot{\boldsymbol{x}} = \begin{pmatrix} 0 & 1 \\ -\omega^2 & 0 \end{pmatrix} \boldsymbol{x}$$

where $\boldsymbol{x} = (\ x \quad \dot{x}\ )^T$. This vector will serve as input in the network. Throughout the process ω is fixed to 2.

B. Training process

For the training process, a neural network architecture with three hidden layers was utilized. Therefore $L = 5$ as it includes inputs layer and output layer. The first two hidden layers contain 20 hidden neurons each, while the final hidden layer consists of 32 neurons. The approach was tested using three different setups. Firstly, the model was evaluated under the condition where the time step were kept constant that means :

$$\exists \Delta t \in \mathbb{R}_+^* \quad \forall i \quad \Delta t_i = \Delta t$$

Subsequently, the model's performance was assessed under variable time step. These two testing scenarios were employed to comprehensively evaluate the model's robustness and effectiveness across different temporal configurations. Such an approach provides insights into the model's ability to generalize and adapt to varying temporal resolutions, offering a thorough assessment of its performance under diverse conditions.

Both of these configurations were implemented using the TensorFlow framework. The training problem is solved by using the stochastic gradient descent method Adam [2] with a decaying learning rate of 0.001.

C. Constant time step data

In the constant time step configuration, the dataset was generated using the Runge-Kutta 45 method implemented in SciPy. The time span ranged from 0 seconds to 10 seconds with increments of 0.1 seconds. The initial condition at $t = 0$ is set to $(x, \dot{x}) = (1, 0)$. The model is trained over 3500 epochs.

D. Variable initial conditions data

In the second configuration, the dataset was also generated using the Runge-Kutta 45 method, but this process was repeated 50 times. For each iteration, a new frequency $f \in [\![0, 60]\!]$ of points is chosen such that $\Delta t = 1/f$. Initial conditions are also randomly fixed between -25 and 25. Once this process is completed, the dataset consists of 150,000 points. Subsequently, the dataset is shuffled. The model is trained over 200 epochs using batches of 500 points each.

E. Time-Exclusive Input

As mentionned in section B one approach of the PINNs is to use only one input, which is t, given the state at $t = 0$. This is equivalent to the case where $\Delta t_i = t$ and $t_i = 0$ for all i resulting in $u_{t_i} = u_0$. All other parameters remain exactly the same as in section C.

IV. Results

A. PINNs compared to simple NN

First, let us compare PINNs and simple NNs - that means $\lambda_1 = 1$ and $\lambda_2 = \lambda_3 = 0$ (see Eq (3))-. In figures 1 and 2 the red curves and green curves represent the predicted x and $\dot{x}$ respectively. The red dots and green dots represent the derivatives of x and $\dot{x}$ using AD. Finally, the black curves represent the ground truth, which are not visible in the plot as they are hidden by the predictions. The predictions are made on the training dataset which is not relevant here. The important observation is that with the simple NN, the shape of the derivative of the vector $\boldsymbol{x}$ calculated using AD does not match to the shapes of the curves x and $\dot{x}$ which makes the NN quite opaque. However using the PINN the curves appear more physically plausible. The $\dot{x}$ calculated by AD matches the prediction, as it should; the alignment between the red dots and the green curve demonstrates this consistency. This simple example underscores the significance of PINNs in physics, as they can yield more interpretable predictions.

B. Time-Exclusive Input Analysis

As shown in figure 3 one of the problems encountered when using such a PINN to solve our problem, and many others, is that the predictions can slowly diverge from the real solution beyond and even within the training time span. In the equation (10) the null solution is valid; however, it fails to satisfy the initial condition. This divergence can be attributed to the coefficient λ_2 being too high, indicating that the PINN is not robust enough. Hence, in the next subsectionsn, the input is modified to $(\Delta t_i, u_{t_i})$, thereby mitigating the problems associated with large values of Δt.

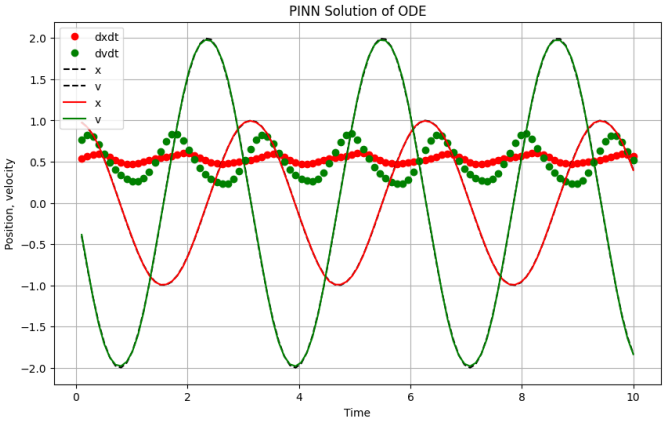


Figure 1. Prediction of the NN on the training dataset

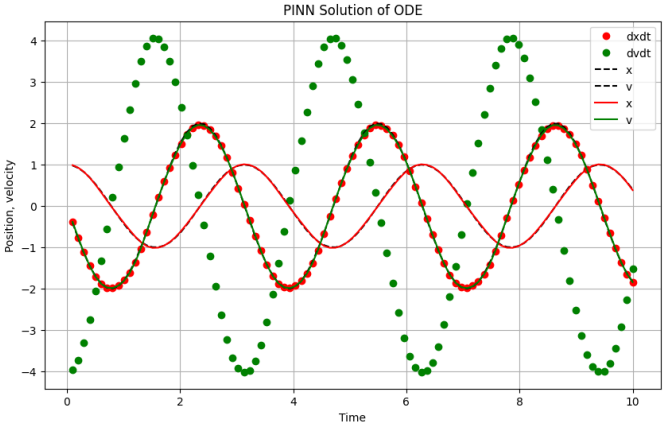


Figure 2. Prediction of the PINN on the training dataset

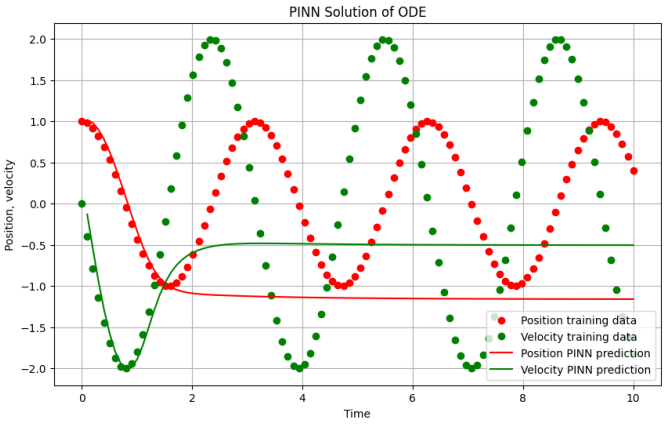


Figure 3. Prediction converges beyond the time span

C. Time and State Vector Input: Enhancing PINN Predictions

In this subsection, the time step are constant, which means $\Delta t_i = \Delta t$ for all i. The PINN is able to predict $u(t)$ very accurately for all t even when the time span during training is restricted to $[0, 10]$. Furthermore, the prediction remains stable when modifying Δt provided it is lower than the value used during training (see figure 4).

One of the drawback is the prediction of $u(t)$ knowing t_0, when $t - t_0 \gg \Delta t$. It requires processing every $\hat{u}_{t_0+k\Delta t}$ until $\hat{u}_t$ as it is done with the Runge-Kutta methods. However as shown in [4] using NN is significantly faster than classical Runge-Kutta methods, up to 100 times faster.

Another problem is due to the dataset. The neural network struggles to generalize when the amplitude of the solution differs from one, as the dataset used during training consists of solutions of (10) with $(x, \dot{x}) = (1, 0)$ implies that the amplitude of u is one. Furthermore, as illustrated in figure 5, the derivative calculated by AD no longer coherent with $\boldsymbol{x}$. That leads to the last trained PINN in this article.

By incorporating solutions of (10) with various amplitudes, the neural network now demonstrates improved generalization capabilities, although not flawless. For instance in figure, when utilizing the initial condition $(x, \dot{x}) = (-35, 55)$ which lies outside the training boundary, the results are only approximate. However, noteworthy is that the derivative calculated by AD remains consistent; there is observable alignment between the red dots and the green curve.

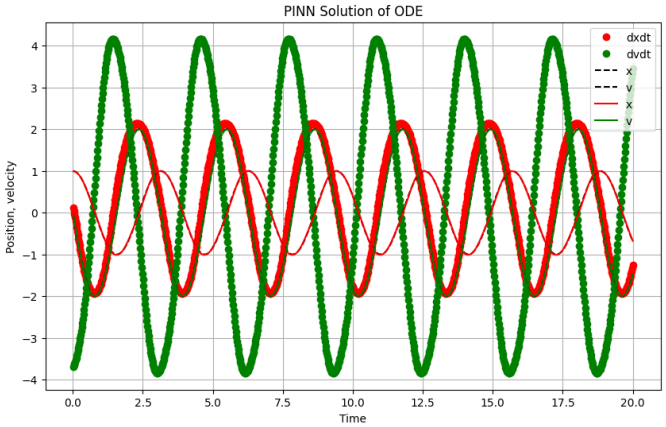


Figure 4. Robustness of the PINN despite fixed Δt during training

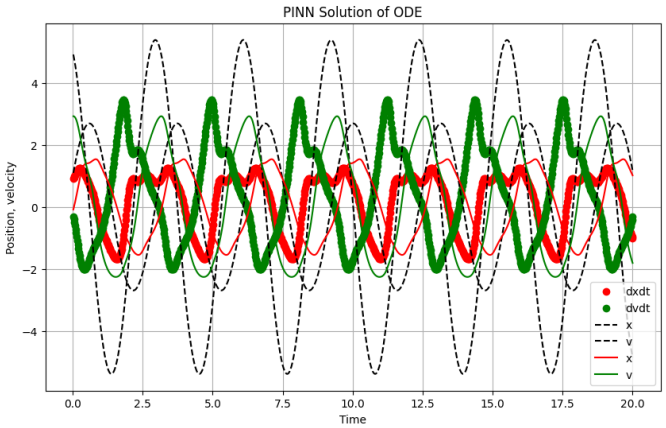


Figure 5. Challenges in Generalization with Varying Amplitudes

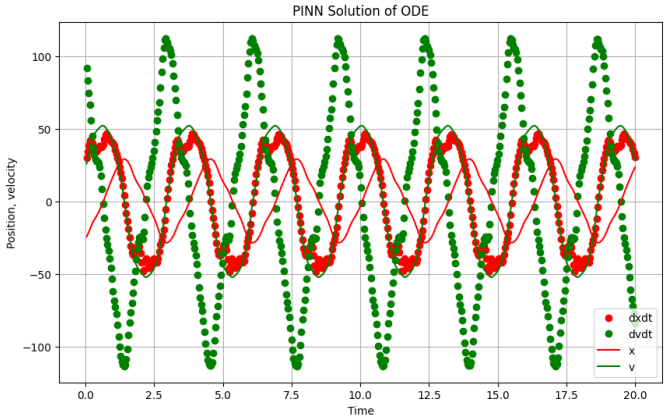


Figure 6.

V. Conclusion

In conclusion, this study explored the application of Physics Informed Neural Networks in solving ordinary differential equations. Through experimentation and analysis, we investigated the performance of PINNs in various configurations, including cases where time steps were constant or variable. We demonstrated that while PINNs offer promising capabilities in accurately predicting solutions to ODEs, they are not without limitations. Challenges arise in cases where the amplitude of the solution differs from the training data and when attempting to generalize beyond the training boundaries. However, by carefully selecting and diversifying the training dataset, we observed improvements in the network's ability to generalize. Despite these challenges, PINNs show significant potential in providing interpretable predictions and fast computations compared to traditional methods. Future research could focus on addressing the limitations identified in this study and further enhancing the robustness and efficiency of PINNs in solving complex physical problems.

Bibliography

[1] Dimitrios Chaniotis and MA Pai. Model reduction in power systems using krylov subspace methods. *IEEE Transactions on Power Systems*, 20(2):888–894, 2005.

[2] Diederik P Kingma and Jimmy Ba. Adam: a method for stochastic optimization. *ArXiv preprint arXiv:1412.6980*, 2014.

[3] Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.

[4] Jochen Stiasny, Samuel Chevalier, and Spyros Chatzivasileiadis. Learning without data: physics-informed neural networks for fast time-domain simulation. In *2021 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pages 438–443. IEEE, 2021.

[5] Xuemeng Zhang, Yaosuo Xue, Shutang You, Yong Liu, and Yilu Liu. Us eastern interconnection (ei) model reductions using a measurement-based approach. In *2018 IEEE/PES Transmission and Distribution Conference and Exposition (T&D)*, pages 1–5. IEEE, 2018.

Image Representation as a Graph for Pedestrian Detection and Tracking

Catherine Rizk
Ensta Bretagne
Brest, France
catherine.rizk@ensta-bretagne.org

***Abstract*—In the field of computer vision, the detection and tracking of individuals play a crucial role in various applications, especially in robotics. This paper presents a comprehensive approach to individual detection and tracking based on skeletonization and graph representation. Initially, the object of interest is isolated using segmentation techniques, followed by skeletonization to extract relevant features. A graph is then constructed based on the skeleton, where nodes represent key points and arcs capture spatial relationships. The proposed method achieves robust individual representation, facilitating shape matching and recognition tasks.**

***Index Terms*—graph, skeleton, detection, vision**

I. Introduction

The pursuit of individual detection and object tracking stands as a central focus within contemporary computer vision research, finding widespread utility across various domains, notably in robotics where precise object tracking is frequently indispensable. When aiming to detect a person, multiple attributes can be examined, including color, intensity, texture, and shape. Yet, shape emerges as particularly pertinent in image interpretation. For instance, the human brain adeptly discerns objects through simple silhouettes, disregarding color and texture in favor of well-defined outlines. Nonetheless, leveraging shape-based criteria for computer vision-based pedestrian detection encounters a significant challenge: the vast diversity in human shapes and sizes. This paper introduces an approach to build robust descriptors for individual detection and tracking by extracting features from a human skeleton representation and transforming it into a graph.

II. State of the art

The field of computer vision has witnessed significant advancements in individual detection and tracking techniques in recent years. Traditional methods often relied on handcrafted features and heuristic algorithms, which limited their performance in complex environments. However, with the advent of deep learning, there has been a paradigm shift towards data-driven approaches that leverage large-scale annotated datasets and powerful neural network architectures.

One of the most prominent techniques in individual detection is the region-based convolutional neural network (R-CNN). These methods achieve remarkable accuracy by combining region proposal generation and object detection in a unified framework. Additionally, single-shot detectors (SSDs) and You Only Look Once (YOLO) architectures offer real-time performance with competitive accuracy.

In the domain of object tracking, tracking-by-detection approaches have gained popularity due to their effectiveness in handling object occlusions and appearance changes. Methods such as the Multiple Object Tracking (MOT) framework and deep learning-based trackers like DeepSORT and SORT achieve state-of-the-art performance in multi-object tracking scenarios.

Despite these advancements, challenges persist in individual detection and tracking, particularly in scenarios with cluttered backgrounds, varying illumination, and occlusions. Addressing these challenges requires robust feature representations and sophisticated algorithms capable of handling complex visual environments.

Skeleton-based methods have emerged as promising alternatives for individual detection and tracking. By representing objects as skeletons composed of key points and structural relationships, these methods offer a compact and informative representation of object shapes. Techniques such as skeletonization and graph-based approaches enable the extraction of meaningful features from skeletons, facilitating tasks such as shape analysis, motion estimation, and behavior recognition.

Skeletonization is also an effective method for obtaining robust descriptors for machine learning, as it captures essential structural information while reducing the complexity of the object representation. This approach facilitates the extraction of meaningful features that are invariant to scale, rotation, and translation, making it suitable for various machine learning tasks.This is particularly evident in the approach proposed in the document [1], which suggests a skeletonization based on stereovision. It then utilizes these skeletons to construct graphs that serve as descriptors for implementing an SVM.

In this context, our proposed approach leverages skeletonization and graph-based representation to achieve robust individual detection and tracking. By combining structural information from skeletons with spatial relationships captured in graphs, our method offers a promising solution to address the limitations of existing techniques. Experimental results demonstrate the effectiveness of our approach in challenging real-world scenarios, highlighting its potential for practical applications in computer vision.

III. Human representation as a graph using distances mapping

A. Pretreatment

1) Object extraction: The following algorithms were implemented on an isolated object on an image with a plane background. To extend this method on any type of images, the object whose graph we want to created must be extracted of the original image. One of the prevailing techniques in image analysis is the k-means algorithm. This method involves partitioning an image into k distinct clusters and assigning each pixel to a cluster by minimizing the distance between the pixel and the centroid of the cluster it belongs to. The k-means method has the disadvantage of requiring prior knowledge of the number of classes to be instantiated. Numerous other methods that are more efficient and better suited to our problem exist today. The reference "Object Detection and Tracking via Active Contour-based Region Segmentation" by Wassima Ait Fares in [2] proposes a method known as active contours, which involves drawing a closed contour around the object of interest and gradually evolving this contour until it perfectly fits the object's outline. This technique is particularly useful for tracking, as the contour evolves over time and can therefore follow the object as it moves. The principle described by Wassima Ait Fares involves defining the active contour segmentation, incorporating energies such as edge, region, and curvature energies to guide the contour towards the object boundary while maintaining smoothness and coherence. By combining the flexibility of active contours with well-defined energy criteria, this approach provides a robust and adaptable method for precise object detection and tracking in computer vision applications.

2) Image segmentation: Now that the object is isolated on a monochrome and uniform background, we proceed with its segmentation to obtain a monochrome shadow with distinct contours. We start by converting our image to grayscale and applying a Gaussian filter to blur the image. This will help us reduce the contrast in certain areas and thus make our single-colored object more easily discernible. Then, we detect the contours using the Canny method and dilate these contours. Next, we fill all the pixels enclosed within the contours with a single color to obtain an image representing a completely black object against a white background. This preparatory step lays a solid foundation for subsequent analysis and further processing.

Fig. 1. Original image

Fig. 2. Post treatment image

B. Creation of a distances map

The next step involves constructing a distance map to add texture to our object, enabling us to trace its skeleton. The work of Aurélie Leborgne [3] describes the algorithm for generating such a map. The map is created from a wavefront that propagates at a constant speed from the shape's boundary towards the interior. Pixels reached simultaneously have the same height and form a contour line. Consequently, we obtain a ridgeline figure (points where the distance is maximal), which we will use for implementing our skeleton, as suggested by Leborgne. Next, we utilize the gradients of the obtained image to determine the ridges that will constitute the skeleton of our object. The figure below illustrates the result obtained through such an approach.

Fig. 3. Original image

Fig. 4. Distance map

C. Skeleton identification

Creating the skeleton is easily accomplished using the distance map. We retain only the areas of maximum gradient and trace all the obtained points. This method preserves both the skeleton and the contour of our initial shape. To eliminate this contour, we utilize the contour of the object shape obtained during preprocessing and subtract it from our current image. This yields a skeleton devoid of the object contour. However, the resulting skeleton is often too thick. To refine it, we perform an erosion operation on the image, which

effectively nibbles away at the skeleton's extremities, making it thinner. Visually, we observe that the obtained skeleton serves as a potent descriptor in computer vision. It enables distinguishing the individual's shape by eliminating numerous variable properties such as thickness and color, thus preserving a common structure across all individuals. This characteristic facilitates the establishment of a recognition criterion for individuals. To facilitate further manipulation, the final step involves converting the skeleton into a graph.

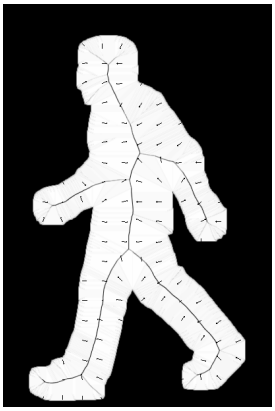


Fig. 5. Gradients around a ridge

Fig. 6. Skeleton obtained from the gradient

Fig. 7. Graph nodes obtained from the skeleton

D. Graph construction

Now, we proceed to construct a graph containing information about the skeletons. The graph is built in a manner similar to the process described in reference [1]: nodes are either placed at the ends of the skeleton branches or at intersections, and arcs connect two nodes without passing through a third node.

1) Nodes identification: To obtain nodes, we go through certain pixels from the skeleton's points list. As the skeleton is quite dense, we can afford to scan the list with a step of 10. We then consider all other points different from the one being studied in the scanned list, also with a step of 10, and count the number of points contained within a circle centered in the first point. Therefore, each point is associated with the number of points contained within its circle, and we retain only those with the highest count. This method allows us to obtain a set of fairly relevant points but it still contains too many elements. If we increase the minimum threshold of the number of points required to be included in a circle for a point to be retained, we empty certain areas of all their points, such as the ends of the arms, for example, which contain far fewer points than the torso. Therefore, we choose to perform a filtering instead. Many of the selected points are close to each other and could be replaced by a single point. We decide to rescan our lists similarly with the points selected in the first step and to draw a new circle around these points. All points contained within a circle are then replaced by a point whose coordinates are the average of the coordinates of all points. Thus, we obtain a list of points that seems to represent the desired nodes in a relevant way.

2) Arcs identification: To determine the arcs, we rely on the nodes and information from the skeleton. In the nodes list, for each node, we identify the three nearest nodes, and for each of them, we determine the segment connecting them to the node under consideration. We then count the number of points in this segment that are included in the set of skeleton points and retain the point with the highest score. This ensures that the obtained arc is close to one of the branches of the skeleton. After this step, most of our arcs are drawn and seem highly relevant, but some nodes that should be connected are not. To resolve this issue, we consider the nodes that are connected to only one other node and seek to connect them while ensuring not to connect nodes at the ends. We reevaluate for each point in the list if there is a path connecting it to another point in the list that contains numerous pixels included in the skeleton. This method allows us to obtain the missing arcs and thus finalize the construction of our graph. Each time an arc is determined, we record the two endpoints of this arc along with their distance. Eventually, we obtain a complete graph describing the structure of a human body.

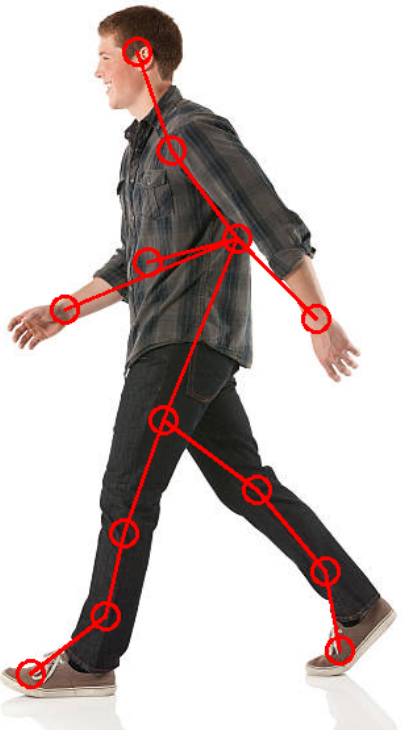


Fig. 8. Graph nodes and arcs obtained from the skeleton

IV. Results analysis and possible applications

The initial objectives have been achieved, as we have succeeded in obtaining a graph representing an individual. The main drawback of this technique lies in its relatively high complexity. In addition to this algorithm, segmentation algorithms should be employed to extract the image of the individual before the presented algorithm must be applied. The uses of this graph are manifold. Graphs are particularly easy descriptors to manipulate and represent an object in a relevant manner. In the reference [3], Leborgne demonstrates that skeletons can be used for matching and shape recognition. Graphs are also particularly useful for tracking, as they simplify the problem of object tracking into tracking nodes and arcs of a graph. Finally, graphs constitute robust and suitable descriptors for machine learning. This is notably demonstrated by Frédéric Suard, Alain Rakotomamonjy, and Abdelaziz Bensrhair in [1] where graphs are employed as descriptors for an SVM, enabling the establishment of an individual detection algorithm.

References

[1] F. Suard , A. Rakotomamonjy, A. Benrhair, “Détection de piétons par stéréovision et noyaux de graphes,” Laboratoire PSI (Perception, Systèmes d’Information), CNRS FRE 2645, INSA de Rouen, France

[2] W. Ait Fares. “Détection et suivi d’objets par vision fondés sur segmentation par contour actif basé région,” Automatique / Robotique. Université Paul Sabatier - Toulouse III, 2013. Français. NNT : . tel-00932263

[3] A. Leborgne. “Appariement de formes basé sur une squelettisation hiérarchique,” Autre [cs.OH]. Université de Lyon, 2016. Français. NNT : 2016LYSEI075 .tel-01694036

Simulation of excavator control using reinforcement learning

Louis Roullier
ENSTA Bretagne
Brest, France
louis.roullier@ensta-bretagne.org

***Abstract*—This article presents a general method to compute an end effector tracking trajectory using reinforcement learning. The approach is applied to an hydraulic excavator arm. Due to the high non linearity of the system, people can't really provide precise mathematical equations or an analytic model. That is why we chose to use a neural network model trained on measurements collected during operations. The distances between joints is only required to set up a simulation to train a control policy using reinforcement learning. The outputs of this process represents the commands directly applicable to the machine without specific post-processing. In our context, the method is tested on a 323 hydraulic excavator without interaction with the soil. The performances of the learnt controller are quite good compared with commercial grading controllers which require hand tuning by engineers and trigger potential difficult adaptation on a specific system.**

***Index Terms*—robotics, excavator, reinforcement learning, control, hydraulic system**

I. Introduction

Mobile earth-moving machines are widely used in the construction sector. Indeed, vehicles as excavators and tracks enable to easily realize landscaping operations (trenching, leveling...). However, these machines are frequently employed in challenging outdoor environments. Therefore, the operations done with these machines can lead to injury and require very-skilled operators. This is why many manufacturers are exploring automation solutions for these machines.

The aim of almost each excavation operation is to change an environment characterized by diversity and variable geometry. Therefore, the main challenge of these operations is to model the environment/machine physical interaction. Due to this difficulty to set complete and fixed model of the environment, people can not use traditional methods with mathematical fixed and precise equations to describe the situation.

Specifically, the objective is for the autonomous machine to adapt to various external factors and make decisions based on the task and environmental conditions. Therefore, machine learning methods such as reinforcement learning can be considered as an option. Indeed, we want to take action in a particular environment and optimize them by maximizing a reward defined by the user.

The automation of machines can be done by several ways : training a fuzzy logic algorithm to imitate the behavior of a skilled-human operator, a tele-operated assistant system can control a real excavator via a wireless local area network...

Machine learning models have been more and more used in automation during the past years. An example from the area of automation of heavy machinery is a study examining the application of Proximal Policy Optimization (algorithm that trains a computer agent's decision function to accomplish difficult tasks) to teach an agent to operate an excavator by training a neural network.

The objective of this study is to computationally combine efficient simulations with the reinforcement learning algorithm. This combination enables the creation and training of an autonomous agent. It will enable to ensure more security and efficiency in the operations.

II. State of the art

A. Robotics Control

The performance of a controller in robotics is measured by the error between the current desired position P_{d}(t) and the real current position P(t). An efficient control law aims to reduce this errors as fast as possible by minimizing deviations and overshoots.

Here are introduced the main ways to control an excavator. First, people can use a Proportional Integral Derivative (PID) in which people use the error e(t= = P_{d}-P(t), its integral and its derivative. The expression for the command is given by :

$$u(t) = K_p e(t) + K_i \int e(t)\,dt + K_d \frac{de(t)}{dt} \tag{1}$$

where: $u(t)$ is the control command, K_p is the proportional gain, K_i is the integral gain, K_d is the derivative gain. The gains can be determined empirically by the user. The main advantage of PID controllers is their simplicity to implement.

An articulated arm can also be controlled in terms of articulated positions α(the angles of the arm) or in terms of end-effector positions P (cartesian coordinates). The conversion between the derivatives of these expressions is given thanks to a Jacobian matrix :

$$\dot{X}(t) = J\dot{\theta}(t) \tag{2}$$

B. Reinforcement learning

[3] summarizes well the main principles of reinforcement learning. In the context of machine learning, there are three main models : supervised learning, unsupervised learning and

reinforcement learning. In our document, we will use the third one. The way it works is represented on the following figure :

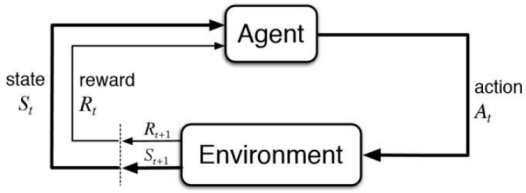


Fig. 1. Reinforcement learning structure

Fig 1 show that there is an agent which interacts with an environment. The agent is an entity which takes decision and interacts with an environment. It provides action in the environment using a policy. This policy is a rule for selecting actions and can be stochastic (it gives a probability distribution over actions) or deterministic (maps to an action). In function of the impact of the action on the environment, the agent will receive a scalar reward defined by the user and the state at t time. The goal is to implement actions which will maximize the reward through time. In order to keep the Markovian propiety of the problem, the observation is extended to several time steps.

C. robotics control using reinforcement learning

In 2020, [4] presented a two-step methodology: 1. Learning the simulation of the excavator and 2. Training a controller with Trust Region Policy Optimization (TRPO) using the previously learned simulation. The method is validated on a real excavator moving the bucket to perform leveling at the ground contact limit. [1] extends this work by providing a more detailed documentation of the methodology and transitioning from position control to velocity control using Proximal Policy Optimization (PPO).

III. Method

The general method is divided into two main steps. First, we model the actuation using measurements collected during operation of the machine. Then, we train an end effector velocity tracking controller in simulation using reinforcement learning with PPO algorithm.

A. description of the machine

In the context of our work, we will use a Caterpillar excavator 323. This robot has 6 degrees of freedom : left and right track, boom, stick and bucket (elements of the articulated arm of the excavator). The excavator is represented on figure 2 :

Here, C_{i} for i=1,2 or 3 represents the command of the three actuators, X_{e} represents the distance between the bucket's middle teeth and the center of the track, X_{j} is the distance between the center of the tracks and the basis of the bucket. Finally, Z_{e} is the altitude difference between the center of the excavator and the bucket's middle teeth and Z_{j} is the altitude

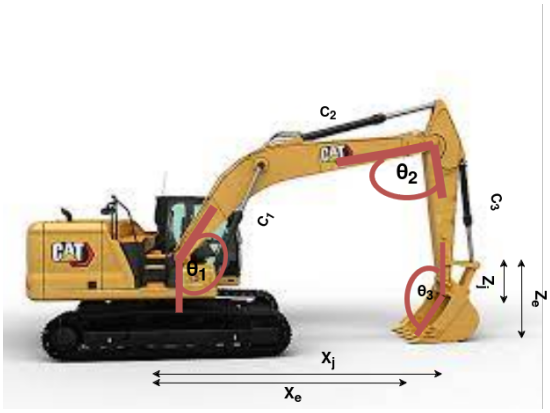


Fig. 2. Representation of the Caterpillar Excavator 323

difference between the center of the excavator and the basis of the bucket. To drive the excavator, the operator has several joysticks : A left one which serves to turn the cab around the tracks group and a right one to pilot the arm (represented by the boom, the stick and the bucket). In our context, we will focus on moving the arm.

B. Modelization of the actuation

In this section, the very first task is to collect data during operations on the machine. The data should contain joints positions, joins velocities and commands at different times. In order to collect data, an operator can design trajectories (circle, curves...) with the end effector. In our case, we can simulate an articulated arm drawing 700 circular trajectories with Python [5]. For each trajectory, we chose a random radius and a random center for the circle. This enables to have joints positions and joints velocities. A photograph of the data collection is provided on the following figure :

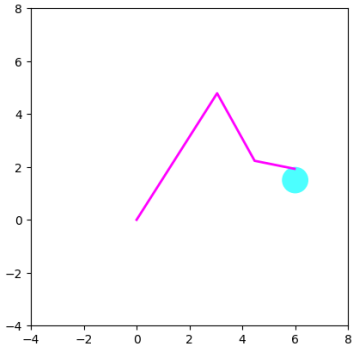


Fig. 3. Data collection using Python

In order to get the commands, we use models developed by [6] which rely a joint velocity to the associated valve command (a scalar between -1 and 1).
The goal of this section is to create a neural network which predicts joints velocities at time t+0.01s from the joint positions at time t and an history of the commands and the joints

velocities. The inputs and the outputs are compiled in Table 1 :

TABLE I
ACTUATOR MODEL INPUTS AND OUTPUTS

Inputs
Joint Positions $q_j(t)$
Joint Velocities $\dot{q}_j(t)$, $\dot{q}_j(t-0.01s)$, ..., $\dot{q}_j(t-0.1s)$
Valve Setpoints $u_j(t)$, $u_j(t-0.03s)$, ..., $u_j(t-0.99s)$
Outputs
Joint Velocities $\dot{q}_j(t+1)$

In order to find the best hyperparameters for our model, we used a Python library called Optuna. This tool is designed to do gradient descent but enables to have global minimum instead of local mminimas. In our case, we give Optuna a list of hyperparameters and for each trial, the library builds a neural network with hyperparameters chosen in the given list. The goal is to minimize the mean-squared error between the neural network predictions and the test set. The mean-squared error function is :

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3}$$

The hyperparameters we chose to design are : the dropout rate, the number of hidden layers, the hidden layers size, the learning rate and the activation function. Given that the process takes a long time, we will search the best hyperparameters within 80 trials.
Finally, the graph which gives the logarithm of the MSE according to the number of trial is given on the following figure :

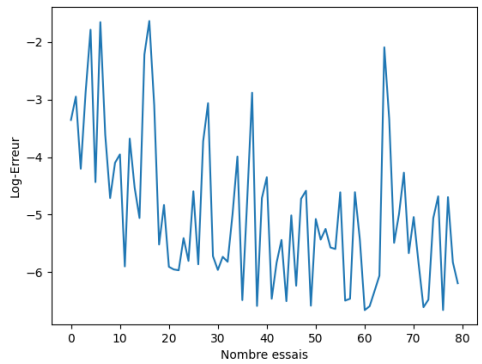


Fig. 4. Logarithm of the MSE between neural network predictions and the test set

Thanks to this graph, we choose a neural network with 1 hidden layer of size 512, a dropout rate of 0.285509, a learning rate of 0.0001. In addition to that, the activation function is tanh. Finally, people can see that the prediction of joints speeds at time t+1 is very precise. In order to have the joints positions, we can just compute an integration with a very known model such as Euler or Runge-Kutta.

C. Calculation of the forward kinematic

The next step is to calculate the parameters X_{e}, X_{j}, Z_{j} and Z_{e}. We also calculate the angle between the teeth and the horizontal axis and the angle between the bucket and this same axis. The determination of these values should be as precise as possible. That is why we use a URDF model for the robot. The URDF (Unified Robot Description Format) model is a standardized XML-based file format commonly used in robotics to describe the physical and kinematic properties of a robot. It provides a comprehensive representation of a robot's structure, joints, sensors, and other relevant information. The URDF model of the excavator 323 is given on the following figure :

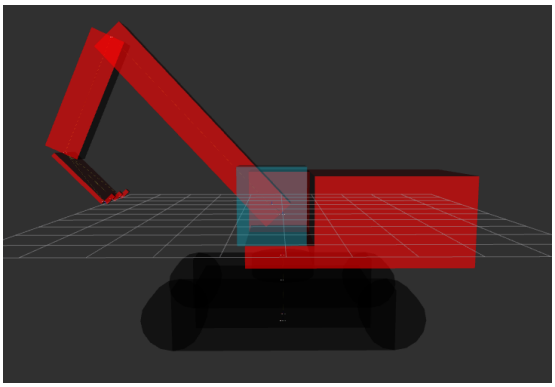


Fig. 5. URDF of the excavator 323

From this URDF, we can easily calculate transforms between different frames. Indeed, the tf2 ROS2 node enables to compute the translations and rotations between the different elements of the robot. These transforms will be used to calculate Z_{e}, Z_{j}, X_{e}, X_{j} and the angles mentioned previously. Nevertheless, in order to be very precise, it is really necessary to calibrate the URDF. To do that, people can drive the excavator during a little amount of time. During the operation, people can acquire some data which should contain the joint states at time t, the position of the cab at time t and the real position of the teeth at time t. We used an IMU system to have the pose of the cab and we took GNSS antennas to have the position of the teeth. After the operations, we determined the best parameters for the URDF using Optuna. We wanted to minimize the MSE between the real teeth positions and the teeth positions provided by the URDF.

D. Learning a Velocity Tracking Controller in Simulation

The goal is to obtain a controller which can track trajectories in task space relative to the cabin frame with the teeth of the bucket.We solve this problem using RL, because it provides a means of dealing with the highly nonlinear actuator dynamics by learning the optimal control inputs through trial and error. To facilitate training, we decouple linear and angular motion. Without decoupling, careful reward tuning is required such that the agent learns tracking both, linear and angular, velocities. The general method is

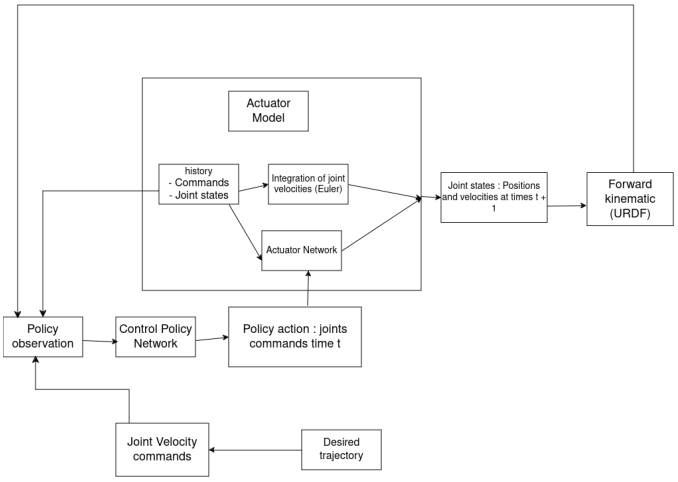


Fig. 6. Diagram of the general method of velocity tracking controller

summarized in figure 6. In this picture, people can see how the elements are relied. The desired trajectory is an arbitrary one decided by the user. The speed commands relatives to the desired trajectory are given by the following formulas :

$$\dot{X}_{j_{\text{command}}} = k_{p_x} \cdot (X_{\text{jtraj}} - X_j) + \dot{X}_{j_{\text{traj}}} \tag{4}$$

$$\dot{Z}_{j_{\text{command}}} = k_{p_x} \cdot (Z_{\text{jtraj}} - Z_j) + \dot{Z}_{j_{\text{traj}}} \tag{5}$$

$$\dot{\alpha}_{j_{\text{command}}} = k_{p_\alpha} \cdot (\alpha_{\text{jtraj}} - \alpha_j) + \dot{\alpha}_{j_{\text{traj}}} \tag{6}$$

Where α_{j} corresponds to the angle between the bucket and the horizontal axis.
As mentioned previously, we used Proximal Policy Optimization (PPO) to have the valve commands at time t. It is a reinforcement learning algorithm that iteratively refines an agent's policy to find an optimal strategy in a given environment. At its core, PPO employs a surrogate objective function to guide policy updates, integrating a clipping mechanism to limit large changes in the policy during each iteration. This clipping ensures stability and prevents the policy from diverging too significantly from the previous iteration. Concretely, it enables to have relative continuous movements for the excavator arm. There will not be a sudden jump of the arm which could be really dangerous for people in the environment during operation. PPO typically performs multiple optimization epochs on collected data, utilizing a value function to estimate expected cumulative rewards and calculate advantages for policy updates. Operating within an actor-critic framework, PPO strikes a balance between exploration and exploitation, facilitating robust learning across various environments. In our case, the reward function will be given by the following formula :

$$r_k = \max\left(0.0, r_v^k + \dot{r}_\alpha^k + r_r^k + r_c^k\right), \tag{7}$$

$$\begin{aligned}
r_v^k &= 0.05 \exp\left(-60 f_{c,1} \|\mathbf{Xj}_k^* - \mathbf{Xj}_k\|_2^2\right), \\
\dot{r}_\alpha^k &= 0.02 \exp\left(-60 f_{c,1} \|_k^* - \dot{}_k\|_2^2\right), \\
r_r^k &= -0.75 f_{c,2} \|\mathbf{a}_k - \mathbf{a}_{k-1}\|_1, \\
r_c^k &= 0.025.
\end{aligned}$$

Where, a_{k} represents the valve command at the interaction k with the environment. This reward function has been given by [1]. Once we defined the reward, the observations and the actions (valve commands), we can design the control policy network : In our context, we take a neural network with tanh as an activation function and a size of hidden layers of 128. Finally, in order to evaluate the results, people can compute the error between reference values and the learning values. In our case, we took the joints velocities norms. We had a mean-error superior to 0.2 rad/s. The results we wanted to have are computed in the following figure extracted from [1] :

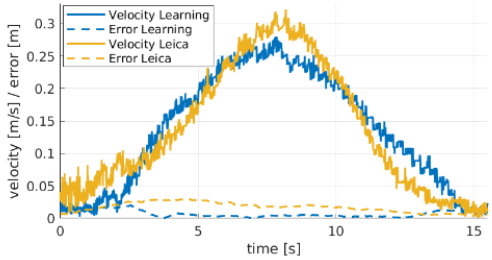


Fig. 7. Error between the learning velocity and the wanted velocity

Here, Leica corresponds to the speeds sensor installed on the machine. Our error can come from several things. First, it would have been better to have real data and not simulated one. Secondly, the coefficients were not really adapted to the situation. Then, the parmeters for the control policy network are maybe not optimal.

IV. Conclusion

This work shows a method to automate the arm of a heavy hydraulic excavator for high-precision grading using a data- driven actuator model that captures the nonlinearities of the hy- draulic actuation and a control policy trained in simulation using RL that directly outputs pilot stage valve commands without requiring joint level gain tuning by an expert. Moreover, users don't really need high knowledge of the machine which is quite interesting. It also means that the process can be applied to other excavator types. The results provided in the study are quite interesting. Indeed, we can predict very well the joints velocities and positions at time t+0.01s. Moreover, the forward kinematic calculation is very precise thanks to the calibrated URDF. Finally, it would be very interesting to take data with a real machine and to implement the process with PPO on a real excavator.

REFERENCES

[1] P. Egli and M. Hutter. “A General Approach for the Automation of Hydraulic Excava- tor Arms Using Reinforcement Learning”. In: IEEE Robotics and Automation Letters 7.2 (2022), pp. 5679–5686.

[2] M. Abd Elmoumen DJABALLAH. ”Système de prédiction de la consommation d’énergie basé Deep Learning”.

[3] T. Pouplier, B. Brousseau-Rigaudie, M-A. Dumont, ”Apprentissage par renforcement appliqué à des robots bipèdes”

[4] P. Egli and M. Hutter. “Towards RL-Based Hydraulic Excavator Automation”. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2020, pp. 2692– 2697.

[5] L. Jaulin ”robmooc”. p 13

[6] J.Brugiere ”Contrôle Robotique Autonome avec Apprentissage par Renforcement Profond” p 12

[7] I. Kurinov, G. Orzechowski, ”Automated Excavator Based on Reinforcement Learning and Multibody System Dynamics”. In IEEE Access (Volume: 8) pp. 213998 - 214006

Probabilistic consensus decision making algorithm for robotic swarm

Etienne Roussel

[a]*ENSTA Bretagne, Brest, France, Mars 2024*

Abstract

This paper introduces a consensus algorithm for swarms of basic agents, such as robots equipped with constrained sensing, processing, and communication capabilities. The algorithm addresses collective decision-making within a network of interconnected robots. In this context, decisions are treated as abstract choices, thus the algorithm can be used for a broad range of applications with specific decisions. Each robot within the swarm is considered as a probabilistic finite state machine, with preferences for a set of discrete states defined as a probabilistic mass function. Then, the individual preferences are updated via local negotiation with directly connected robots.

1. Introduction

A robotic swarm aims to achieve physical and computational flexibility and increased system robustness in multi-robot tasks, such as localization, mapping, and navigation in an unknown, possibly dynamic, environment. The main characteristics of a swarm robotic system includes the following [1, 2]:

- Autonomy: Robots are autonomous. Individual robots and the entire swarm exhibit different levels of autonomy.
- Localized sensing and communication: Each robot's sensing and communication capabilities are local.
- Decentralized control: Individual robots do not have access to centralized control and global knowledge.
- Cooperative action: Robots cooperate with each other to perform an intended task.

These characteristics uniquely qualify robotic swarms to perform certain types of tasks effectively, such as large area coverage within a short time; tasks in dynamic, uncertain, or unstructured environments; tasks that require scaling up or down; and tasks requiring redundancy in information[1].

Despite the potential benefits, the application of swarm robotic systems to real-world challenges faces various engineering dificulties. One major obstacle involves achieving swarm behavior without centralized control. Previous research has concentrated on attaining specific global swarm behaviors by implementing relatively straightforward rules at the individual robot level. The targeted global behaviors encompass swarm aggregation [3], shape/pattern formation [4], and navigation [5]. To effectively address real-world problems, a robotic swarm must demonstrate the capability to sequentially execute multiple global behaviors. For instance, successfully localizing and retrieving an object from a confined tunnel may necessitate the swarm to aggregate, adopt a linear formation, navigate, and collaborate in transporting the object back to the designated base location.

This paper is based on the precedent work of Yang Liu of a decision making algorithm for swarm robots [1]. So this paper presents his algorithm. The problem is formulated as a consensus decision making process given a finite number of choices for individual robots. The objective of the algorithm is to acheive a global consensus among the robots, it is not to aggregate or navigate. To do so, individual robots are modeled as Probabilistic Finite State Machines (PFSMs), where their finite states are defined by a set of executable distinctive behavioral rules but these are abstract here. A successful global behavior emerges when the majority, if not all, of the robots execute the same rule-set simultaneously. Since local sensing/communication and decentralized control are assumed, decision making must also take place at the individual robot level while consensus in the individual decisions is sought for achieving a global behavior at the swarm level. Each robot's preference towards n possible choices is defined as a Probability Mass Function (PMF). The choice with the highest preference is called the "exhibited decision" of the robot. The presented method aims to achieve consensus based on local communication between the nearby robots and internal processing of the individual preferences.

1. Modeling individual robots as PFSMs by generating initial preference distributions over given choices;

2. Updating each robot's preferences based on its own and locally connected robots' preferences; and

3. Accelerating convergence and conflict resolving by increasing confidence toward the exhibited decision.

The presented algorithm focuses on achieving guaranteed consensus over a finite set of abstract decisions for a group of robots in a single network. It achieves consensus in a swarm network regardless of its connectivity density, i.e.,

consensus can be reached when the network is fully connected or even when sparsely connected. But all the robots have to be connected to the network, i.e to be close enough to each other. Given a fully connected network condition (i.e., each robot communicates with every other robots), a simple majority rule would be sufficient, and the presented algorithm would fall back to the same majority rule. However, if the network is sparsely connected and only highly localized communication is available, the presented algorithm would more effectively resolve the conflicting decisions within the network than the methods based on the majority rule. The robots in majority rule based methods aim to gather direct information from as many other robots as possible, which is difficult in a sparsely connected network. By targeting consensus in a standalone network, the problem setup and algorithm are well positioned for further designing complex sequential swarm robot behaviors.

PFSM-based modeling methods have been widely applied for different swarm formation problems, such as aggregation [3] and chain formation [6]. Biological inspiration has also played an important role in many swarm decision making algorithms. The psychology of a human group can inspire as well for swarm applications [7].

2. The Algorithm

The presented method focuses on achieving consensus in a swarm of simple robots, and thus the following constraints are considered:

- Individual robots are primitive with limited sensing, communication, and processing capabilities.
- Communication in the swarm is local; each robot can communicate only with nearby robots within the communication range.
- Robots have no temporal memory (i.e., no log of history data) and function like finite state machines.

It is further assumed that the network topology does not change during the decision making process. If the decision making process is relatively fast enough compared to the robots' physical movements, the change in the network topology would remain trivial while the physical locations of the robots may change during this process.

The overall collective decision making algorithm, consists of three parts: (1) initializing the swarm network with random preference distributions; (2) updating individual preference distributions based on local interactions; and (3) improving confidence in the decision. Each node representing a primitive robot follows these steps until a consensus in the swarm is reached.

2.1. Initialization of the swarm network

Let $\mathbf{R} = \{1, 2, \ldots, m\}$ be an index set of m robots in the swarm and $\mathbf{Q} = \{1, 2, \ldots, n\}$ be the index set of n distinct choices, corresponding to global swarm behaviors. Individual robot is referred to as R_k for $k = 1, \cdots, m$. Each robot's preference towards n choices is modeled as a Probability Mass Function (PMF), such that $\sum_{j=1}^{n} P_k(j) = 1$, where $P_k(j)$ indicates the R_k's preference toward the choice j. This probability distribution is hereafter referred to as the preference distribution. Each robot exhibits one decision at a time, determined by the corresponding index of $\max\{P_k(1), \cdots, P_k(n)\}$. Initial values of these preference distributions are randomly generated for the initial set up of the swarm network.

2.2. Preference updating via local interaction

Each robot updates its own preference by interacting with its neighboring robots within the communication range. If a robot is too far, the swarm has to aggregate before starting the decision making process. For R_k, all neighboring robots of R_k and itself forms a local connection group, denoted by C_k. Each robot holds IDs of all members within its local connection group. Robots within the same connection group exchanges their preference distributions. A local consensus group, D_k is defined as a non-empty set of the robots connected to R_k that express the same decision as R_k. Each robot within a local consensus group shares information of the IDs of all members, but not their the preference distributions. When a robot decides to join or leave a local consensus group, its closest neighbors in the group can detect the change, and this information is broadcasted within the local consensus group. So each robot have access to the number of robots directly and indirectly connected with the same decision in addition to the preference distribution of the directly connected neighbors only.

Figure 1 shows an example of how these two groups are determined for a network of 8 robots. The node colors indicate the exhibited decision of the robots. When two connected robots exhibit the same decision, the connecting lines are also visualized with the color of the decision. For R_4, the local connection group is defined by C_4; and the local consensus group showing the "red" decision is D_4. As shown in the figure, members of D_4 may not be directly connected to R_4, but forms a connected network including R_4. It is noted that R_4 shares IDs and preference distribution with the members of C_4 while sharing only the IDs with the members of D_4.

Each robot updates its own preference distribution by taking account of preferences of other robots in its local connection group:

$$P_k(j) = \frac{\sum_i N_i P_i(j)}{\sum_i N_i}, \quad i \in C_k, j \in \mathbf{Q} \tag{1}$$

where $N_i = |D_i|$ is the size of the R_i's local consensus group. This equation is a weighted average of the preference distributions among all directly connected robots, where the weights are determined by the size of the decision group. In the equation, each robot compromises its preference by considering its neighbors' preferences, where the decision agreed by more robots carry a larger weight in this process. If C_k is a subset of D_k, then the decision has been locally converged at R_k, and (1) results in an equally weighted average. The weights proposed

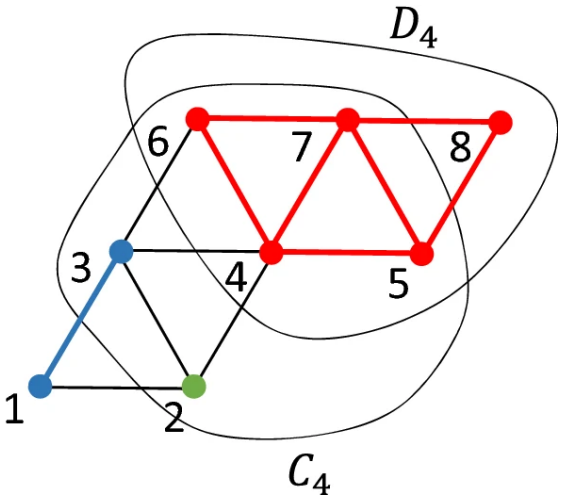


Figure 1: A network of 8 robots, showing R_4's local consensus group C_4 and local connection group D_4

in (1) help resolve potential conflicts among all the local consensus groups by favoring large-sized local consensus groups.

2.3. Internal processing for decision uncertainty reduction

Once the following two conditions are satisfied, R_k is considered locally converged:

- (1) local consensus is achieved (i.e., $C_k \subset D_k$).
- (2) the maximum difference of the preference distributions within C_k is below a threshold.

The maximum difference in the preference distributions among the members of C_k is defined as λ_k and calculated by

$$\lambda_k = \max_{k1,k2 \in C_k, k1 \neq k2} \left(\sum_{j=1}^{n} \mid P_{k1}(j) - P_{k2}(j) \mid \right). \quad (2)$$

λ_k is a measure of the degree of divergence in C_k. If $\lambda_k < \lambda_T$ is satisfied for an empirical threshold value λ_T, then R_k is considered being confident about its own exhibited decision.

Once the above conditions are satisfied, R_k's preference distribution is further updated to accelerate the convergence process by multiplying a linear multiplier, $\mathscr{L}_k(j)$, for $j \in \mathbf{Q}$:

$$P_k^{new}(j) = P_k(j)\mathscr{L}_k(s(j)) \quad (3)$$

where s(j) is introduced to rearrange the probabilities in P_k in a descending order, such that $P_k(j)$ is in s(j) position in the new order. $\mathscr{L}_k$ is constructed as follows:

$$\mathscr{L}_k(j) = L_l \frac{j-1}{n-1} + L_u \frac{n-j}{n-1}, \quad j \in \mathbf{Q} \quad (4)$$

where $L_\ell = \frac{1}{n}\left(\frac{\lambda_k}{\lambda_T}\right)^{0.3}$; $L_u = \frac{2}{n} - L_\ell$. L_ℓ and L_u are the lower and upper ends of the linear multiplier. This process reduces the uncertainty on the exhibited decisions of individual robots by increasing the preference values of the highly preferred choices and further reducing the preferences values of the less preferred choices.

Thus, Figure 2 shows a random 100-node network ($m = 100$) and its convergence towards a consensus using the algorithm described above. Different colors indicate exhibited decisions of individual nodes among 30 possible choices ($n = 30$). An edge between two nodes turns into a specific color if they form a local consensus group, D_k.

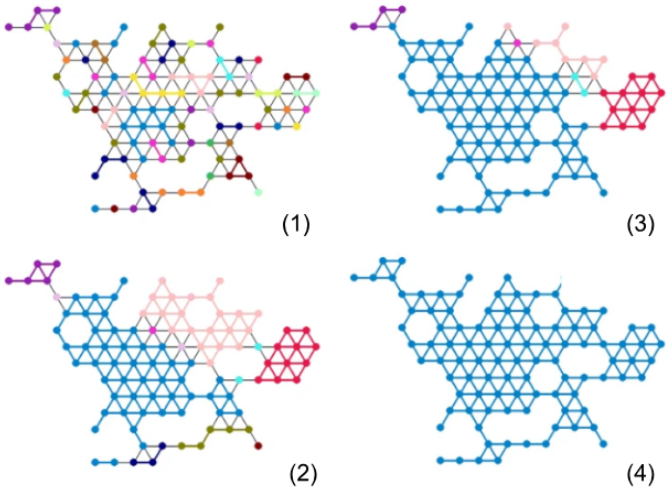


Figure 2: The evolution of the consensus process for 100-size network

3. Conclusion

A new distributed collective decision making algorithm for swarm robotic applications has been presented in this paper. Individual robots are assumed to be primitive with limited sensing, communication, and processing capabilities. Under this assumption, while individual robots may exhibit any of n possible decisions, the swarm can only exhibit a global behavior if most of individual robots, if not all, agree on a unique decision. The presented algorithm achieves guaranteed consensus in a connected network.

References

[1] Liu, Y., Lee, K *Probabilistic consensus decision making algorithm for artificial swarm of primitive robots*, SN Appl. Sci. 2, 95 (2020).
[2] M. Brambilla, E. Ferrante, M. Birattari, M. Dorigo, *Swarm robotics: a review from the swarm engineering perspective*, Swarm Intell, 7(1), 1–41 (2013).
[3] O. Soysal, E. Şahin, *Probabilistic aggregation strategies in swarm robotic systems*, In: Swarm intelligence symposium, 2005. SIS 2005. Proceedings 2005 IEEE. IEEE, pp 325–332 (2005).
[4] J. Yang, X. Wang, P. Bauer, *Line and V-shape formation based distributed processing for robotic swarms*, Sensors 18(8), 2543 (2018).
[5] F. Ducatelle, G.A. Di Caro, C. Pinciroli, F. Mondada, L. Gambardella, *Communication assisted navigation in robotic swarms: self-organization and cooperation*, In: 2011 IEEE/RSJ international conference on intelligent robots and systems. IEEE, pp 4981–4988 (2011).
[6] S. Nouyan, A. Campo, M. Dorigo, *Path formation in a robot swarm*, Swarm Intell 2(1), 1–23 (2008).

[7] M. Moussaid, S. Garnier, G. Theraulaz, D. Helbing, *Collective Information Processing and Pattern Formation in Swarms, Flocks, and Crowds*, (2009).

Improvement of Underwater Image Quality through a Machine Learning Approach

Mathys SÉRY
ROB24 - STIC
ENSTA BRETAGNE
Brest, France
mathys.sery@ensta-bretagne.org

***Abstract*—In the context of our projects in Robotics (ROB) at ENSTA Bretagne, we have encountered systems where the core of the project revolves around vision. However, in the case of underwater robot projects, vision becomes a real challenge due to the unique conditions presented by such environments. The perfect example, which is the focus of my project this year, is the Guerlédan Lake. Upon observing the visual results from the robot's camera in this setting, I questioned the existence of more efficient algorithms that differ from our traditional histogram equalization algorithms (already used for this project). The most interesting solution I found was developed by two individuals: Cheol Woo Park and Il Kyu Eom. Their solution involves the use of a neural network that takes into account the impact of the aquatic environment on the distribution of the R, G, and B channel histograms of color images. Their solution appears to be adaptable to various underwater environments. However, despite the promising results presented in their article, I obtained opposite results after a personal implementation attempt. The likely causes of this difference stem from the disparities between our two computing configurations: due to a lack of computational power, I had to reduce my dataset and the number of epochs for training my model.**

***Index Terms*—Underwater image enhancement, Convolutional Neural Network, Normalization, Standardization, Robotics**

Fig. 1: Applications of vision in robotics

I. Introduction

The use of computer vision, especially in the context of robotics, is a versatile tool full of potential. In classical applications, one can find obstacle detection, object classification, as well as mapping and localization with methods such as SLAM [1].

Robotics evolves in different environments and in different ways, from robotic arms to flying drones to autonomous cars (fig. 1). In the case of our robotics studies at ENSTA Bretagne in ROB, we are led to explore robotics in marine/aquatic environments. This branch of robotics has characteristics of its own in terms of dynamic behaviors of systems, sensors, and especially from a vision perspective. Aside from factors that can affect the appearance and quality of the image stemming from the system itself (quality of equipment, information storage method, etc.), we have factors specific to aquatic environments: variations in brightness depending on depth, the turbidity of the environment, water quality, the presence of suspended particles in the water, etc. [2], [3].

To overcome these challenges, one can employ general image processing algorithms or specialize in underwater imaging to improve and restore the quality of our images and/or videos taken underwater. Techniques for underwater image restoration focus on correcting alterations using physical models, or image enhancement techniques aiming to increase the visual quality of the image with more varied and less precision-oriented approaches [4]. Within these different categories, a wide variety of techniques can be found, for example, in the frequency domain, spatial domain, or color constancy (=human ability to identify the color of an object independently of its lighting conditions) [3].

But in recent years, machine learning and the tools it develops have been massively used in various fields, especially to modify and/or improve images. These tools

seem to have a promising future with good results [1], [4], [5]. Therefore, in this research exercise, we will focus on one of the many developed convolutional neural networks, which we will analyze and compare with other more traditional methods.

II. Analysis of the Chosen Method

To construct this small work, I relied on and focused on an article written very recently and published in 2024 by Cheol Woo Park and Il Kyu Eom [5]. In this article, they describe and use the following network schema (fig.2).

The main architecture of this neural network is composed of four major blocks in different quantities. We have:

- K + 1 ASNet (Adaptive Standardisation Network) blocks,
- 2 ANNet (Adaptive Normalization Network) blocks,
- 1 convolutional layer,
- 1 sigmoid activation function.

Let's now discuss the utility of each of these blocks and the organization of this network.

A. Classical Functions: Convolutional Layer and Sigmoid Activation Function

These two functions are present both at the end of the main architecture and in the internal architecture of an ASNet or ANNet block, which is why it is useful to explain them first.

1) Convolutional Layer: The convolutional layer has several possible uses in a CNN: it allows extracting important features from an image or reducing the dimension of the image while preserving important data. A convolutional layer can exploit the spatial aspect of the data values provided at its input, making it effective with data like images. Placed at the end of our overall architecture, we ensure that the global model's output is an image with 3 channels, while the number of channels increases during the model.

2) Sigmoid Activation Function: Traditionally, this function is used for binary classification tasks where the output should be interpreted as a probability. Here, it normalizes the output values to a fixed range, thereby controlling the output values. In the article, this part has been minimally documented.

B. ASNet Block

In natural images, the chromatic distributions of blue, green, and red tend to coincide. There are no significant differences between each channel. However, in the case of underwater images, the distribution of the red channel is shifted towards lower pixel values (recall the range of values a color pixel can take is from 0 to 255). Therefore, to improve our image, we need to align the histograms even in this scenario.

A classic solution that is applicable is the standardization technique. This technique allows achieving this alignment by resizing the features of our image using statistical tools such as mean and standard deviation.

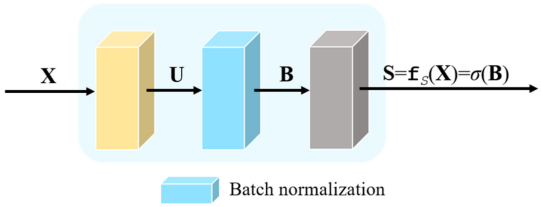


Fig. 3: ASNet Architecture

The ASNet block was designed by the two authors to implement this technique and thus match these histograms. According to Figure 3, it is composed of:

- 1 convolutional layer,
- 1 Batch Normalization (BN) block,
- 1 sigmoid activation function.

The BN block is responsible for the standardization of features. It also allows for a higher learning rate and less meticulous initialization. This same block also handles the adaptive part of the ASNet block:

At the input of the Batch Normalization block, we have an input $u = [u_1, u_2, ..., u_i, ..., u_N]$, and at the output, $b = [b_1, b_2, ..., b_i, ..., b_N]$. For each b_i, we have,

$$b_i = \alpha_i * \frac{u_i - \mu(u_i)}{\sqrt{\text{Var}(u_i) + \epsilon}} + \beta_i$$

where α_i and β_i are adjustable parameters, ϵ is a very small value to avoid division by zero without impacting the operation too much, μ correspond to a mean value.

As standardization gives unbounded results, we use the sigmoid function to restrict the range of possible values. However, this causes saturation for extreme values, which can affect the quality of the final image.

C. ANNet Block

This second block, conceived by the two authors of my reference article, contributes to achieving satisfactory performance in color restoration and contrast enhancement. To accomplish this restoration and improvement, a normalization technique is employed. This technique involves modifying the range of values of input features by stretching the contrast based on the minimum and maximum values of these features.

However, this technique is sensitive to outliers. In this case, it is necessary to use the minimum and maximum values

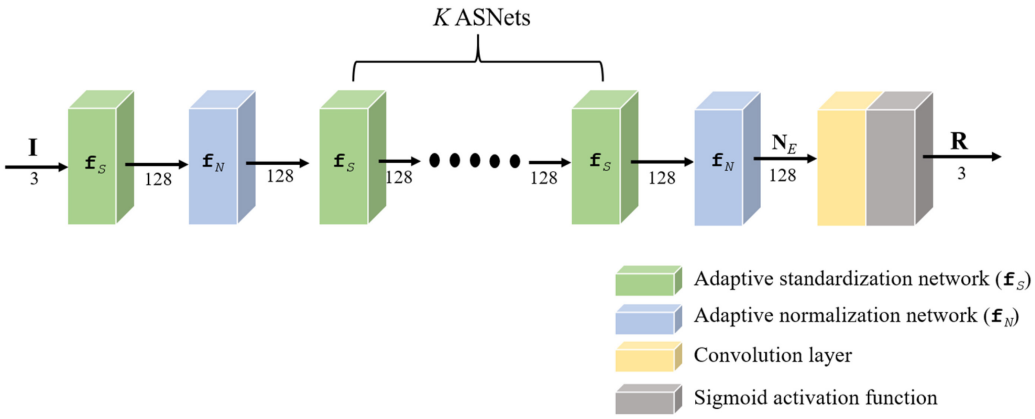


Fig. 2: Proposed CNN model by Cheol Woo Park and Il Kyu Eom(subject of the study)

in percentiles. For a given input feature X, we obtain a normalized version X_{nor} as follows:

$$X_{\text{nor}} = \frac{X - \text{minp}}{\text{maxp} - \text{minp}}$$

where maxp (resp. minp) corresponds to the maximum (resp. minimum) value in percentiles. Generally, this technique is ineffective on underwater images due to the diversity of their chromatic distributions. However, it is possible to achieve interesting results by finding suitable values for min and max in percentiles. The challenge with the normalization technique is that it is difficult to find optimal values as they differ for each given feature.

To address this issue, the authors have developed a block named ANNet, allowing for a sophisticated selection of extreme percentile values that can adapt to each given feature (Adaptive), resulting in excellent normalization (Normalization).

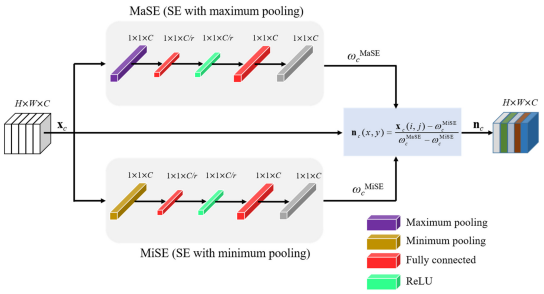


Fig. 4: ANNet Architecture

According to Fig. 4, its architecture contains two parallel blocks: one MaSE block and one MiSE block, each providing the maximum and minimum values in percentiles. For any input feature X, we have the normalized feature X_N as follows:

$$X_N = \frac{X - \min_p}{\max_p - \min_p}$$

MiSE and MaSE have an almost identical architecture. They consist of two Squeeze and Excitation (SE) blocks to which we add, at the very beginning, the search for a maximum (max pooling) or a minimum (min pooling).

The SE block aims to adaptively recalibrate the responses of each feature per channel, taking into account inter-channel connections. The first operation (Squeeze operation) aims to compress the spatial data of input features using a global average pooling operation: calculation.
The second operation (Excitation operation), composed of two Fully Connected (FC) layers separated by a ReLU function and followed by a sigmoid function, aims to capture all inter-channel dependencies. It exploits the information aggregated during the previous operation.

4. Global Architecture

Even though each block of the neural network seems to have interesting individual properties, it is only by combining them in a specific order that truly interesting results are obtained:

- At the very beginning of the network, the first ASNet allows obtaining the initial standardized features from our input image and aims to align the distribution of these features.
- As the values output by ASNet can be too saturated for extreme values, it is followed by an ANNet block to address these shortcomings. This block adaptively stretches the distribution of features, ultimately assisting in color restoration and improving the contrast of the final image.
- The subsequent K ASNets extract a sufficient number of interesting features and progressively align the distributions of these features. According to their tests, Cheol Woo Park and Il Kyu Eom observed that K=5 ASNets

seemed to be the best compromise for obtaining a high-quality image at the output of the overall network.
- Through their ablation tests (keeping and/or removing certain parts of the neural network to study the output), they observed that using 2 ANNets leads to a more ”natural” improvement in the enhanced output image. Hence, there is a second ANNet block after the K ASNets. These tests highlighted the importance of ASNet in color correction and ANNet in contrast enhancement.

III. Experimental results

To assess the performance and outputs of this neural network, I aimed to compare it with other more traditional methods. I also employed the algorithm used in our Guerlédan project, named ”ESPADON,” which utilizes the CLAHE algorithm, brightness correction, and another algorithm applying histogram equalization followed by CLAHE on each channel of the base image [6]. This entire program was executed via Google Colab to train the model and immediately compare its results with others.

Note: The ”ESPADON” project is a third-year ROB 24 project led by Titouan BÉLIER, Jules LE GOUALLEC, Mathieu PITAUD, and Mathys SÉRY, supervised by Thomas LE MEZO. The images from this project come from the camera of a BlueROV, an ROV from BlueRobotics.

To align more closely with the training in the article, I kept exactly the same optimization and loss functions (adam and mean square error), as well as the same learning rate values. Due to my limited computing power, I had to drastically reduce the number of data points taken in training and the number of epochs: moving from training on over 6000 images for 250 epochs to training on 300 images (randomly selected to ensure variety in images and situations) for 100 epochs. With this small comparative test, my goal is to check if my model is usable for the ”ESPADON” project in its current state.

The first comparison is based on a capture taken in the Guerlédan lake during an approach mission to our structure. To navigate, the robot must use the ARUCO markers present on this structure. The obtained result is shown in Fig.5.

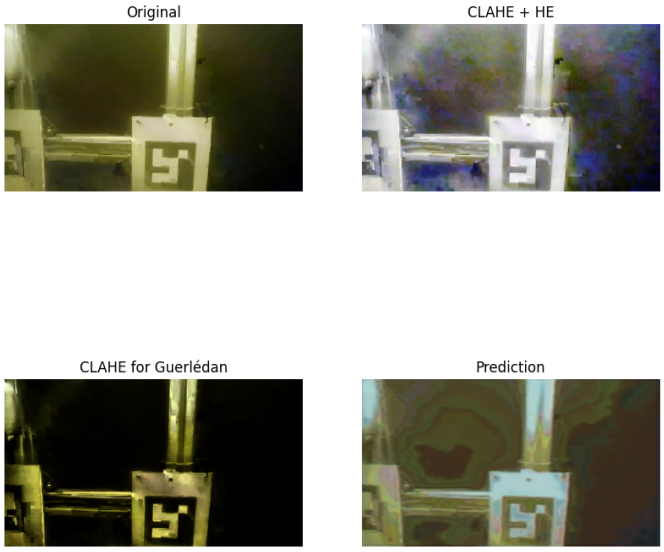


Fig. 5: Comparison of Underwater Image Processing at Lake Guerlédan

The second test is on another capture taken during a pool test at ENSTA Bretagne. We are using the same equipment but in different conditions. The results are presented in Fig. 6.

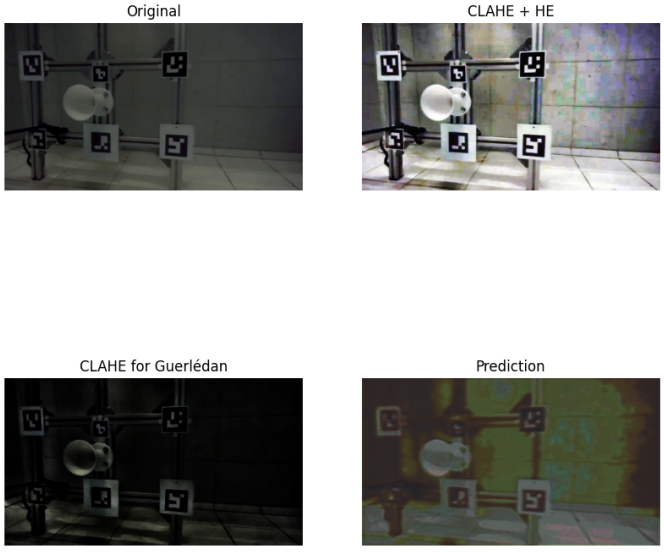


Fig. 6: Comparison of Underwater Image Processing in the Lab-STICC Pool (ENSTA Bretagne)

These two trials highlight the weakness of Machine Learning methods: to achieve a quality result, a large, varied, and high-quality training set is required, along with sufficient computing power to train our model extensively and in-depth. Traditional methods prove to be better choices for processing.

Although the results obtained in the article are very encouraging (Fig.7), they deserve further refinement in

training to find an optimal configuration even under minimal computing capacity.

Fig. 7: Results taken from the reference article [5] of this study

IV. Conclusion

In conclusion, we can assert that this CNN, chosen as a case study, is, in theory, an excellent idea with significant potential. However, at the present moment, employing it for ROV image processing and developing it with common hardware (such as a portable student PC without extensive technical capabilities) seems challenging. This is evident from the results of tests conducted on captures from the "ESPADON" project. Traditional methods, therefore, are not on the verge of disappearing to be entirely replaced by machine learning-based approaches.

References

[1] B. Van Eden and B. Rosman, *An overview of robot vision*, in *2019 Southern African Universities Power Engineering Conference/Robotics and Mechatronics/Pattern Recognition Association of South Africa (SAUPEC/RobMech/PRASA)*, pp. 98–104, IEEE, 2019.

[2] A. Marouchos, M. Sherlock, and J. Cordell, *Challenges in underwater image capture*, in *OCEANS 2018 MTS/IEEE Charleston*, pp. 1–5, IEEE, 2018.

[3] W. Zhang, L. Dong, X. Pan, P. Zou, L. Qin, and W. Xu, *A survey of restoration and enhancement for underwater images*, *IEEE Access*, vol. 7, pp. 182259–182279, IEEE, 2019.

[4] S. Raveendran, M. D. Patil, and G. K. Birajdar, *Underwater image enhancement: a comprehensive review, recent trends, challenges and applications*, *Artificial Intelligence Review*, vol. 54, pp. 5413–5467, Springer, 2021.

[5] C. W. Park and I. K. Eom, *Underwater image enhancement using adaptive standardization and normalization networks*, *Engineering Applications of Artificial Intelligence*, vol. 127, p. 107445, Elsevier, 2024.

[6] Omer Deperlioglu and Utku Kose, *Practical method for the underwater image enhancement with adjusted CLAHE*, In *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*, pages 1–6, 2018, IEEE.

Representation of the Websocket protocol with Colored Petri Nets

Zafrana Joachim

March 1, 2024

Abstract

In today's digital landscape, where instantaneous interaction is not just a luxury but a necessity, WebSockets have emerged as a fundamental technology driving real-time communication on the web. From live chat applications to collaborative editing platforms, WebSockets offer a seamless and efficient way to establish persistent, bidirectional communication channels between clients and servers. In this article, we delve into the world of WebSockets, exploring their architecture, advantages, and diverse applications across various industries. Join us as we unravel the transformative potential of this powerful web technology.

Contents

1 Introduction

Petri nets (PN), or Petri net models, are mathematical and graphical tools used for the modeling and analysis of concurrent systems. They were introduced by Carl Adam Petri in the 1960's and have since found applications in various fields, including computer science, systems engineering, and biology. They can be defined as a graphical structure which contains places, transitions, arcs and tokens. The dynamics of a Petri net model are governed by the movement of tokens between places and transitions according to predefined rules. Tokens represent the state of the system, and transitions represent events or actions that can occur. When a transition is enabled, meaning that it has sufficient tokens in its input places, it can fire, causing tokens to move from input places to output places based on arc weights.

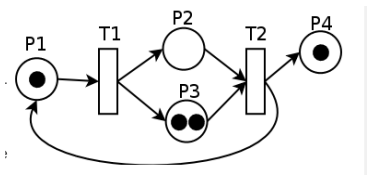


Figure 1: Simple examble of a PN

Since PN allow two tokens to be crossed simultaneously, they are particularly useful for checking the operation of computer protocols (TCP ...). Furthermore, Colored Petri Nets integrate a programming language, typically Standard ML, to model and manipulate data. This programming language allows for the definition of complex data structures, functions, and algorithms within the CPN model. It allowed the development of PetriCode which is used for automatically generating protocol implementations, based on their CPN representation. In this study we will focus on one of these protocols.

On the other hand, Websocket technology is a rather new technology (still under development). As said previously, WebSocket is a bidirectional, full-duplex communication protocol. Once the initial connection is established, both the client and the server can send messages to each other at any time without waiting for a request from the other party. This allows for real-time, low-latency communication.

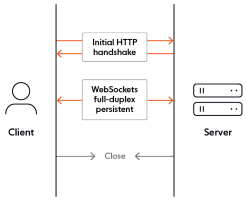


Figure 2: Websocket Protocol [2]

Today, websockets are used extensively in web applications, and their bidirectional communication properties enable them to set up controllers for different systems. For example, in the article *Web-Based control application using Websocket* [3], an application for controlling radiation phenomena is presented. They are also widely used for data visualization, since it's easy to serialize any type of data in JSON format and send it via Websocket, as explained in article *Remote Data Visualization through WebSockets*[4].

The chosen article, taken from *Implementing the WebSocket Protocol Based on Formal Modelling and Automated Code Generation* [1] , shows the interest of PN in the representation and implementation of protocols and more especially Websockets.

2 CPN model for Websocket Protocol

In this article, is showcased an implementation of this protocol using CPN, in order to see the complexity of this protocol.

The following figure illustrates the top-level module of the CPN model representing the protocol system level for the WebSocket protocol. The protocol system comprises two principal entities: the Client and the Server, depicted as substitution transitions labeled accordingly. .

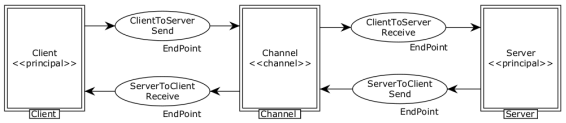


Figure 3: PN representation of the Websocket Protocol

Additionally, the channel module symbolizes the connection between the Client and the Server. It's often host by the server on one of his port, which allows him to communicate with all his clients. Then, below is the Client submodule.

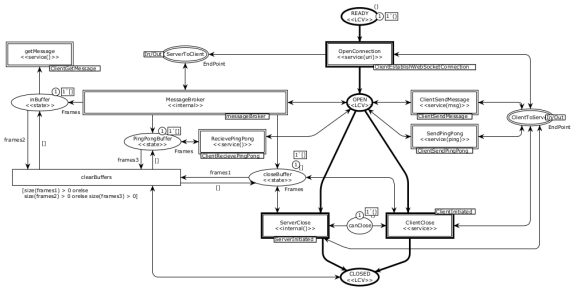


Figure 4: PN representation of the Client Submodule

As you can see, there's a lot of places and transition modules. Initially, the place "READY" contains a token which allows "OpenConnection" transition to be completed, and a token will be placed in the place "OPEN". That's the beginning of the Websocket Protocol, on the client side. Then, all external services will be called, enabling the client protocol to function: keep the connection alive, receive and send messages at any time. Finally, when desired, the client can choose to close the connection. As explained in the article, the complete CPN model contains 136 places and 84 transitions to represent the complete process, so even if it seems be a lot, it just shows how important are CPN in the implementation of this protocol.

3 CPN WS Model Verification

CPN are also usefull for testing the protocol. Indeed, the previous representation of the protocol allows us to explore all reachable states of the model, in order to eliminate those we don't want them to happen, with their potential errors. To illustrate this, the approach adopted in the article based on behaviour properties of the CPN model:

1. Client and Server modules can both open the connection from the initial state. (they have one token each in the place OPEN of their module).

2. All terminal states (without enabled transitions) only happen if the connection has been properly closed.

3. Regardless of the sequence of events, it is always possible to achieve a proper closure of the WebSocket connection from any reachable state.

By using ML methods specific to CPN models (testing all reachable states, according to these 3 properties), we can generate the amount of Nodes and arcs necessary to represent the system. In the case where both Client and Server are sending messages, we saw that we would need approximately 40 000 nodes and 177 000 arcs. We then realised the complexity of the protocol, and that led to the inclusion of new modules and transitions (ClearBufffers), in order to reduce the number of terminal states.

Once a stable and not too complex CPN has been obtained, we can automatically generate code to implement this process using the groovy platform, capable of automatically generating Java code (like it's done in the PetriCode [5]).

4 Possible Websockets applications

Nowadays, Websocket technology finds applications in various real-time communication scenarios, such as Chat Applications, Multiplayer online games, or live streaming applications. In order to showcase its interest, I've developed a Websocket-Based web interface to control a Trax robot whose assiociated URDF representation is presented below.

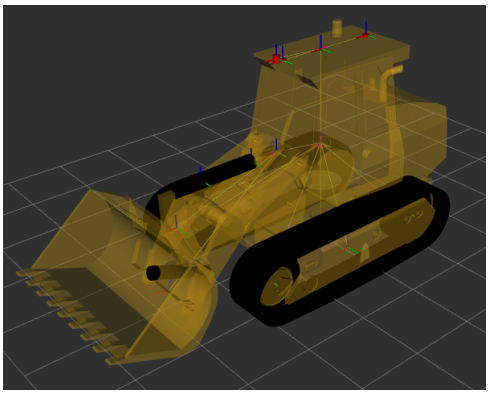


Figure 5: URDF of the Trax Robot

This Trax uses ROS2 as middleware and the point here is to create a link between a Web Interface and the Backend part using a Websocket. Thus, the user of the web interface needs to be able to listen ROS topics, or publish on them. To put in place this Web-Interface, I used the following process :

As Trax is equipped with ROS middleware, we can create a global subscriber in ROS (using python). This subscribes to relevant topics (that we wish to display on the web application), and we generate a dictionary in JSON format that we send every 200ms to the Websocket server via a TCP socket hosted on a local port of the robot's teensy. Once the connection has been opened by a client, this dictionary is sent at regular intervals to automatically update Trax-related data (arm position, current user command, etc.). For its part, the client can also send commands on this Websocket, sent in the opposite direction, to the ROS node, capable of interacting with the robot. This protocol is summarized in the following figure:

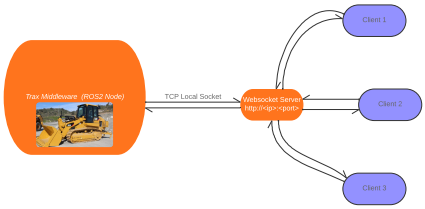


Figure 6: Interfacing process used

Once the WebSocket server has been initialized, the user can connect to it (provided he or she is connected to the same network or uses a VPN), by going to the web page:

`http://<ip-adress-of-the-teensy>:<port-on-which-the-WS-server-is-hosted>`

The user can then control the trax remotely from any device with a web browser.

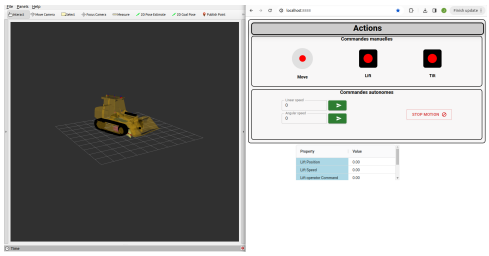


Figure 7: Web interface of the Trax

5 Conclusion

In this study, we explored the field of Petri Nets, and noted their usefulness in the representation of complex processes, such as Websockets. Combined with Petri Code, they enable the automatic generation of JAVA code by AIs that automatically implement these protocols. This greatly facilitates the development of various IT processes, no matter how complex, as long as they can be represented graphically, which is supposedly simpler and more accessible to everyone. By enabling a more robust and structured implementation, this method promotes the reliability and maintainability of IT systems.
Thus, by reducing reliance on manual programming, it paves the way for more efficient and scalable software engineering.

Bibliography

[1] Kent Fagerland Simonsen, Lars Michael Kristensen: Implementing the WebSocket Protocol Based on Formal Modelling and Automated Code Generation

[2] Alex Diaconu: Ably, https://ably.com/topic/websockets

[3] Y. Furukawa: Web-Based control application using Websocket

[4] A. Wessels, M. Purvis, J. Jackson and S. Rahman, "Remote Data Visualization through WebSockets," 2011 Eighth International Conference on Information Technology: New Generations, Las Vegas, NV, USA, 2011, pp. 1050-1051, doi: 10.1109/ITNG.2011.182.

[5] Petri Code : https://github.com/kentis/nppn-cli

Introduction to Traffic Collision Avoidance System, TCAS

Rania ZIANE
ECOLE NATIONALE DES SCIENCES ET TECHNIQUES AVANCEES BRETAGNE
Autonomous Robotics
Email: rania.ziane@ensta-bretagne.org

***Abstract*—Every year, several in-flight airplane accidents occur, resulting in the loss of precious lives. In this paper, we have outlined the operation of the traffic collision avoidance system, designed to prevent collisions between aircraft and thus ensure a safe journey for passengers. We also explored potential improvements to the system, while examining the shortcomings of the current TCAS and the changing needs due to increasing air traffic. Enhancing maneuverability through a new trajectory calculation algorithm and establishing a dedicated communication link contribute to strengthening threat management.**

***Index Terms*—Collision prevention in air traffic, radar and navigation aids, electronic and electronic telecommunications**

I. Introduction

Every year, several airplane accidents occur in mid-flight, resulting in the loss of precious lives [1]. One of the most tragic accidents occurred when a Russian Tupolev flight collided with a DHL cargo plane in mid-air on July 1, 2002, shortly after 11:30 PM, above Überlingen near Lake Constance [2], highlighting the importance of developing collision systems. Numerous studies have focused on this topic, proposing increasingly sophisticated optimizations [3]–[6]. In this document, we proposed a basic model of detection based on the operation of the Traffic Collision Avoidance System (TCAS) [7], designed to prevent collisions between aircraft and thus ensure a safe journey for passengers [8], [9]. We also explored potential improvements to the system, while examining the shortcomings of the current TCAS and the changing needs due to increasing air traffic. Enhancing maneuverability through a new trajectory calculation algorithm and establishing an exclusive communication link contribute to strengthening threat management.

II. OPERATION

A. Basic TCAS

The Traffic Collision Avoidance System (TCAS) operates using advanced radar communication technology, enabling the detection and avoidance of mid-air collisions.

This system sends radar interrogations to nearby aircraft via a 1030 MHz radio frequency and receives responses on 1090 MHz. Through these exchanges, TCAS calculates the relative position of other planes – distance, direction, and altitude – in relation to itself.

Identify applicable funding agency here. If none, delete this.

In case of a detected collision risk, it generates visual and auditory alerts to warn the crew.

If necessary, it also provides precise instructions to alter the flight path, through pitch commands, to avoid the accident. These instructions are coordinated among all TCAS-equipped aircraft involved, ensuring a concerted and safe action. Capable of managing multiple planes simultaneously, TCAS is an essential tool for flight safety, ensuring continuous and proactive surveillance of congested airspace.

Two versions of the TCAS system have been developed. The first version, TCAS I, was enhanced to become TCAS II, introducing several major changes from the original version.

These modifications will be detailed later.

B. The warnings

There are two types of alerts issued by the TCAS system, namely the Traffic Advisory, known by the acronym TA, and the Resolution Advisory, or RA. These alerts constitute the main difference between the TCAS I and TCAS II versions.

When a TA is activated, the pilot is alerted by the display of the word TRAFFIC in yellow on the navigation screen, as well as by a sound announcement saying ”traffic, traffic”.This alert does not indicate the most critical level but mainly aims to draw attention to a potentially conflicting situation. A TA is issued by the TCAS as soon as an aircraft or any unidentified object comes into immediate proximity of the aircraft.

The RA (Resolution Advisory) is triggered if the intruding object continues to approach and the conflicting situation does not improve. Usually, an RA is issued 10 to 15 seconds after a TA. When an RA is activated, the pilot is guided by a vocal announcement instructing them to perform a climb or descent, with instructions on the speed to adopt if required.

According to the description of figure 1

An alarm signal is emitted by the TCAS as soon as an aircraft intrudes into this Tau region, whose threshold is based on the concept of time until the closest point of approach (CPA), calculated by the ratio between the remaining distance and the speed of convergence, both in vertical and horizontal directions.

For example, if the tau time threshold is set at 40 seconds, the alarm is triggered as soon as the intruding aircraft is less than 40 seconds from the CPA. The TCAS system adapts

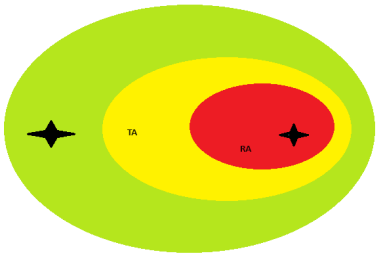


Fig. 1. Intruder aircraft in the TAU.

the tau thresholds according to the current altitude, making detection more sensitive as the aircraft climbs in altitude.

Due to obvious safety concerns, the TCAS system avoids issuing descent instructions when the aircraft operates at a low altitude, just as it refrains from doing so at high altitudes to prevent stalling.

The major distinction between the TCAS I and TCAS II versions is that the former can only generate TA-type alerts (Traffic Advisory), while the latter is capable of producing both TA and RA (Resolution Advisory) alerts. This extended capability is the fundamental reason why TCAS I was upgraded to the TCAS II model.

C. The interrogation protocols

In the field of aviation, transponders play a crucial role by facilitating communication and surveillance between aircraft and air traffic controllers.

To standardize these exchanges, several radio frequency (RF) communication protocols have been established, adapted to different needs and operational contexts. Among these protocols, Modes 1, 2, 3/A, 4, A, C, and S are particularly noteworthy.

Mode S (for Selective) is distinguished by its widespread use in air collision avoidance systems, such as the TCAS (Traffic Collision Avoidance System). Designed to minimize the risks of interrogation overload—a phenomenon that can occur in congested airspaces where many radars attempt to communicate simultaneously with transponders—Mode S allows for a more precise and efficient selection of aircraft to interrogate.

This specificity reduces radar background noise and improves the reliability of transmitted data, thereby facilitating collision detection and avoidance.

Transponders operating in Mode S are also designed to be compatible with Modes A and C.

Mode A relates to the aircraft's identification, while Mode C provides information on its altitude. This interoperability ensures a complete coverage of the information necessary for tracking and managing air traffic.

Beyond its role in TCAS, Mode S is also a central element of ADS-B (Automatic Dependent Surveillance-Broadcast) systems.

These systems allow for the automatic transmission of key information such as the aircraft's identity, position, speed, and direction, without requiring active interrogation by ground radars.

ADS-B messages are emitted at the frequency of 1090 MHz, and a variant of these transmissions can also be relayed by a Universal Access Transceiver (UAT) in the 900 MHz band.

ADS-B significantly improves the situational awareness of pilots and air traffic controllers, contributing to safer and more efficient navigation in global airspace.

D. ADS-B

TCAS equipment capable of processing ADS-B messages can use this information to enhance the performance of the TCAS through techniques known as "hybrid surveillance." In its current implementation, hybrid surveillance utilizes the reception of ADS-B messages emitted by an aircraft to decrease the frequency at which the TCAS equipment interrogates that aircraft.

This reduction in interrogations decreases the use of the 1030/1090 MHz radio channel, which will extend the operational lifespan of TCAS technology over time. ADS-B messages will also provide a low-cost technology for aircraft, delivering real-time traffic information in the cockpit for small planes.

Hybrid surveillance does not involve using the aircraft's flight information in the TCAS conflict detection algorithms; ADS-B is used solely to identify which aircraft can be safely interrogated at a reduced frequency.

This approach allows for more efficient management of airspace by reducing the need for constant radar interrogations, while maintaining accurate surveillance and reducing the risk of saturating the radio frequencies critical for air safety.

III. TCAS SYSTEM COMPONENTS

A. The calculation unit

The TCAS (Traffic Collision Avoidance System) computer unit, or TCAS processor, performs airspace surveillance, intruder tracking, tracking of its own aircraft altitude, threat detection, determination and selection of Conflict Resolution Advisory (RA) maneuvers, as well as generating guidance. The TCAS processor utilizes pressure altitude, radar altitude, and specific aircraft state inputs from its own aircraft to regulate collision avoidance logic parameters that define the protected volume around the TCAS-equipped aircraft. If a tracked aircraft poses a collision threat, the processor selects an avoidance maneuver that will ensure a safe vertical distance from the intruder.

B. The S transponder

The Mode S transponder is an essential component of the TCAS II (Traffic Alert and Collision Avoidance System), playing a critical role in its overall functionality. Without the installation and proper operation of the Mode S transponder, TCAS II cannot operate correctly. In the event of a Mode S

transponder failure, the TCAS Performance Monitor automatically detects this incident and puts the TCAS system into Standby mode, thus ensuring efficient fault management. The Mode S transponder is not only crucial for communication with the ground air traffic control (ATC) system, but it also facilitates the exchange of data between aircraft equipped with TCAS, allowing for precise coordination of collision avoidance maneuvers. A single control panel allows the crew to select and control all aspects of the TCAS equipment, including the Mode S transponder. This panel offers various control options, allowing the crew to choose between different operational modes, including Standby, Transponder, TA only, or Auto/TA-RA, depending on the needs of the situation. In conclusion, the Mode S transponder is a fundamental element of TCAS II, ensuring flight safety by facilitating communication and coordination of collision avoidance actions between aircraft.

C. The antennas

The antennas used by TCAS II consist of a directional antenna mounted on top of the aircraft, as well as either a unidirectional antenna or a directional antenna placed beneath it. Most installations prefer to use the optional directional antenna located underneath the aircraft. These antennas emit interrogations at 1030 MHz with varying power levels in four azimuth segments of 90 degrees each. The antenna mounted beneath the aircraft emits fewer interrogations and at lower power than the one mounted on top. Additionally, these antennas receive responses from transponders at 1090 MHz and transmit them to the TCAS processor. The directional antennas allow for the separation of responses to reduce synchronous interference. Besides the two TCAS antennas, two additional antennas are required for the Mode S transponder. One is mounted on top of the aircraft and the other underneath. These antennas enable the Mode S transponder to receive interrogations at 1030 MHz and to respond to these interrogations at 1090 MHz. The choice between the antenna mounted on top or the one underneath is automatically selected to optimize signal strength and reduce multipath interference. It is important to note that TCAS operation is automatically disabled when the Mode S transponder transmits, to prevent the TCAS from tracking the aircraft to which it is attached.

IV. TCAS II Limits

Despite its advantages, TCAS II has several limitations, but potential solutions could address these issues. First, the TCAS II system can only handle three aircraft simultaneously due to its limited vertical resolutions. Currently, only vertical movements are proposed as a solution, which can be problematic in conflicts at different altitudes. To overcome this limitation, a software update that allows for horizontal resolutions, such as turns or tilts, could be considered to enable TCAS to manage multiple aircraft simultaneously.

Second, contradictory instructions between TCAS and air traffic controllers (ATC) can lead to confusion and potential collisions. Although TCAS generally has priority over ATC, improved communication systems could be envisioned, allowing TCAS to share its resolutions with ATC, so that aircraft receive consistent directives.

Third, the current TCAS focuses mainly on a range-based representation, which may not always be intuitive for pilots. By introducing a time-based representation, TCAS could provide clearer information on imminent collisions by calculating approach rates and announcing the remaining time before collision.

Finally, integrating flight plan information into TCAS could improve its ability to anticipate conflicts. Currently, TCAS relies solely on the detection of intruders without considering the flight plans of the aircraft. Introducing this data would allow for a better assessment of potentially conflicting trajectories, thus enhancing TCAS's ability to avoid collisions.

By adopting these measures, TCAS II could be improved to more effectively meet challenges and enhance safety in air traffic.

V. Modeling and Conflict Management between Aircraft in Flight

Modern aviation faces the critical challenge of ensuring flight safety, particularly by avoiding collisions between aircraft in flight. In this context, the presented simulation aims to provide a modeling tool to study aircraft movement and manage potential conflicts. This paper describes the main aspects of the simulation, including aircraft modeling, conflict calculation, and the utilization of TCAS for conflict management.

A. Aircraft Modeling

An Aircraft class has been developed to represent aircrafts in flight. Each aircraft is defined by its position, velocity, and altitude attributes. Position and altitude updates are performed based on navigation requirements and conflict resolution. Additionally, a method in the Aircraft class allows for gradual adjustment of altitude towards a target altitude to avoid conflicts.

Aircraft 1 speed: 0.139 m/s (approx. 500 km/h converted to m/s and adjusted for simulation time scale). Aircraft 2 speed: 0.133 m/s (approx. 480 km/h converted to m/s and adjusted for simulation time scale). It should be noted that speeds remain constant, unlike altitude, which changes gradually to minimize the risk of collision.

B. Conflict Detection and Management

The time before a potential conflict and the safety distance between two aircraft in flight are calculated based on their relative velocities and positions. An evaluation is then conducted to determine if two aircraft are in a conflict situation, based on specific thresholds of spatial proximity and altitude difference. In the event of a detected conflict, the TCAS system sends altitude change instructions to the involved aircraft to move them away from each other and avoid collision.

La distance entre les deux avions avant l'instruction du TCAS était d'environ 19.46 unités, représentant la distance

euclidienne entre leurs positions dans l'espace de simulation, soulignant leur proximité immédiate avant l'intervention pour éviter un conflit.

C. Results

The simulation results indicate that The initial trajectories of the aircraft in 3D, as well as the evolution of their altitudes over time, are presented in Figures 2 and 3 respectively.

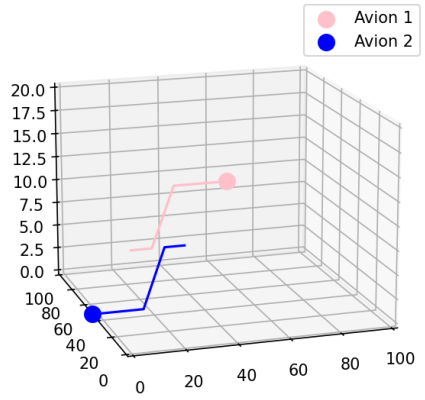


Fig. 2. 3D visualization of trajectories.

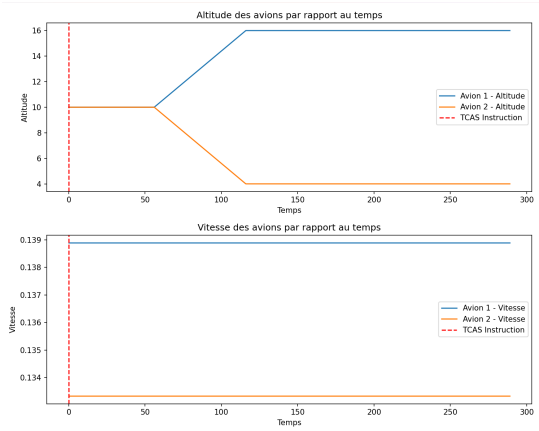


Fig. 3. Altitude and speed graphs

CONCLUSION

The main areas of development in the fight against collision threats are aimed at improving the TCAS SYSTEM. Firstly, it is important to introduce maneuvers combining both vertical and horizontal and horizontal movements. This approach will enable better resolution of threats, especially in dense dense air traffic environments. In addition, the system will need to be updated to transmit the aircraft's flight plan, as well as and other vital navigation data. This additional information will enable a better assessment conflict situations.

Finally, a voice link will have to be established between the two aircraft when they are in the most critical threat zone. These TCAS improvements should considerably reduce the risk of mid-air risk of mid-air collision. However, it is important to recognize that, even with these improvements total prevention of mid-air collisions remains difficult to guarantee due to the complexity of the human factors involved in flight operations.

REFERENCES

[1] Le Tupolev Tu-154, un avion réputé mais victime d'une série noire , Le Figaro. Consulté le: 1 mars 2024. [En ligne]. Disponible sur: https://www.lefigaro.fr/international/2016/12/25/01003-20161225ARTFIG00047-le-tupolev-tu-154-un-avion-repute-mais-victime-d-une-serie-noire.php s

[2] Crash d'Überlingen – Il y a dix ans, le père d'une victime tuait un contrôleur aérien , Tribune de Genève. Consulté le: 1 mars 2024. [En ligne]. Disponible sur: https://www.tdg.ch/il-y-a-dix-ans-le-pere-d-une-victime-tuait-un-controleur-aerien-955096586736

[3] H. Durrant-Whyte, N. Roy, et P. Abbeel, Robotics: Science and Systems VII. MIT Press, 2012.

[4] K. D. Julian, M. J. Kochenderfer, et M. P. Owen, Deep Neural Network Compression for Aircraft Collision Avoidance Systems , J. Guid. Control Dyn., vol. 42, no 3, p. 598-608, 2019, doi: 10.2514/1.G003724.

[5] K. D. Julian et M. J. Kochenderfer, Guaranteeing Safety for Neural Network-Based Aircraft Collision Avoidance Systems , in 2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC), sept. 2019, p. 1-10. doi: 10.1109/DASC43569.2019.9081748. .

[6] A. Calò, P. Arcaini, S. Ali, F. Hauer, et F. Ishikawa, Simultaneously searching and solving multiple avoidable collisions for testing autonomous driving systems , in Proceedings of the 2020 Genetic and Evolutionary Computation Conference, in GECCO '20. New York, NY, USA: Association for Computing Machinery, juin 2020, p. 1055-1063. doi: 10.1145/3377930.3389827.

[7] S. Murugan et A. A.Oblah, TCAS Functioning and Enhancements , Int. J. Comput. Appl., vol. 1, no 8, p. 45-49, févr. 2010.

[8] W. H. H. Iii, TCAS: a system for preventing midair collisions

[9] Enhanced Ground Proximity Warning System . Consulté le: 1 mars 2024. [En ligne]. Disponible sur: https://aerospace.honeywell.com/us/en/pages/enhanced-ground-proximity-warning-system?fbclid=IwAR0x0fTI9a-WjSFfD2LXL7d5zgQWSqxR9CGg7Lv1SiaAu1bh6pSk5DX2pfk