ENSTA BRETAGNE

FISE2024 - Autonomous Robotics and Embedded systems

Multi-sensor calibration of a LiDAR, stereo cameras and thermal cameras. Computer-Aided Design of the system.

Internship report

report defended on September 2023

par BELIER Titouan

UNIVERSIDAD DE MÁLAGA

University supervisor: Dr. Jesús Morales Rodríguez
Academic supervisor: Pr. Luc Jaulin

Acknowledgment

I would like to extend my heartfelt gratitude to the Robotics and Mechatronics Group at the Escuela Técnica Superior de Ingeniería Industrial in Málaga, Spain, for providing me with the opportunity to undertake my internship with them. Special thanks to my supervisor, Dr. Jesús Morales Rodríguez, for his guidance and support throughout the internship.

I am also thankful to the entire team for their collaboration and valuable insights. Their dedication to advancing robotics research has been truly inspiring.

I appreciate the university for providing a conducive learning environment and access to resources that contributed to the success of my work.

Lastly, I would like to express my gratitude to Maïwenn Groleau, a third-year student at École nationale supérieure des techniques avancées (ENSTA) Bretagne specializing in hydrography, for her invaluable knowledge and advice regarding point cloud processing. Her expertise and support have been instrumental in the success of my work.

Figure 1: Experimental day with the Spanish Legion.

Contents

Abstract

This report shows the 3 months ERASMUS internship work in the context of the PREMOVE project [8] at the Escuela de Ingenierias Industriales under the supervision of Dr. Jesus Morales Rodriguez. The goal of this internship was to develop a procedure to calibrate a multi-sensor system integrated by 3D LiDAR, 3 stereo cameras and 3 thermal cameras. The first part shows the components used in this work, how to install their ROS2 packages and how they work. Then,the second part explains how the SolidWorks model has been created to attach the system on top of an autonomous car. Finally, the last part is about how to calibrate stereo cameras to LiDAR and how to improve the visualization.

Résumé

Ce rapport présente le travail de stage ERASMUS de 3 mois dans le cadre du projet PREMOVE [8] à l'Escuela de Ingenierias Industriales sous la supervision du Dr. Jesus Morales Rodriguez. L'objectif de ce stage était de développer une procédure de calibration pour un système multi-capteurs composé de 3 LiDARs, 3 caméras stéréo et 3 caméras thermiques. La première partie présente les composants utilisés dans ce travail, comment installer leurs packages ROS2 et comment ils fonctionnent. Ensuite, la deuxième partie explique comment le modèle SolidWorks a été créé pour fixer le système sur le dessus d'une voiture autonome. Enfin, la dernière partie traite de la calibration des caméras stéréo par rapport au LiDAR et de l'amélioration de la visualisation.

Glossary

CUDA CUDA (Compute Unified Device Architecture) is a parallel computing platform developed by NVIDIA. It enables the acceleration of data processing on GPUs (Graphics Processing Units) using specialized libraries and programming languages. CUDA is widely used to speed up scientific computations and deep learning applications by harnessing the computational power of GPUs.. 13

OpenCV OpenCV (Open Source Computer Vision Library) is an open-source library widely used for image processing and computer vision tasks. It offers a wide range of features for image capture, manipulation, analysis, object detection, shape recognition, and various computer vision applications.. 13

port Porting a ROS1 package to ROS2 involves adapting the package's codebase and dependencies to be compatible with the ROS2 framework, ensuring seamless integration and functionality in the ROS2 ecosystem. This transition typically includes modifying communication interfaces, configuring build systems, and updating libraries for ROS2 compatibility.. 12

RANSAC The RANSAC (Random Sample Consensus) algorithm is a robust method used in computer vision and other fields to estimate mathematical models from noisy or outlier-contaminated data. It works by iteratively selecting random samples, estimating a model, verifying its fit to the data, and selecting the best model based on the number of inliers.. 7, 18, 20

ROS ROS (Robot Operating System) and ROS2 are widely used frameworks that have revolutionized the field of robotics. They provide a flexible and powerful platform for developing, controlling, and integrating robotic systems. ROS offers a rich set of tools, libraries, and community-supported packages, enabling seamless collaboration and rapid prototyping. ROS2, the next generation of ROS, further enhances the capabilities with improved scalability, real-time performance, and support for various platforms. Both ROS and ROS2 empower robotic engineers by simplifying the development process, fostering code reuse, and facilitating the creation of complex robotic applications, making them indispensable tools in the world of robotics. In the following parts of this report we will call ROS2 the new generation of ROS. In this report, the first generation of ROS will be called ROS1 to make the distinction with ROS2.. 6

SolidWorks SolidWorks is a 3D Computer-Aided Design (CAD) software widely used in the engineering and design industries. It allows engineers and designers to create accurate 3D models of parts and assemblies, simulate their behavior, generate technical drawings, and facilitate the design process. SolidWorks is known for its versatility and integration with other design and manufacturing tools, making it a powerful tool for product development and innovation.. 6

Introduction

Context and theme of the internship

With the new progress on self driving, autonomous cars could be a reality in the future of urban areas. However, lots of challenges still need to be solved such as detecting traffic signage and pedestrians. If a human is able to differentiate a greenlight from a redlight, the speed of the car in front of it and to predict the path a pedestrian will take, autonomous cars are not prepared to do it for now. Indeed, different sensors should be used to complete these tasks. A LiDAR can detect 3D shapes and their distances, a stereo camera can also give the colors of the different objects and finally thermal cameras are able to detect heat no matter if it's dark or foggy.

Researchers need to use these different sensors, calibrate each one of them but also calibrate them between each other to have multiple layers of information and choose the right decisions. Finally, a solid mechanical structure need to be done to mount every sensors on top of a car.

From May to July 2023, I had the opportunity to work with the Robotics and Mechatronics Group of the Escuela Superior de Ingeniería Industrial in Málaga, Spain. This group is divided into multiple teams, each team focusing on one particular aspect of robotics.

My internship was a preliminary work of a bigger project Dr. Jesús Morales Rodríguez, my tutor, is working on called PREMOVE. The goal of this project is to develop techniques for detecting and predicting the movement of traffic participants for collision avoidance, which will contribute to the safe integration of self-driving cars in restricted urban areas [7].

Goals of the internship

When I first talked with my tutor about the next months, he told me they needed someone to work on the 3D modeling of their system. As I study autonomous robotics at ENSTA Bretagne, I asked him to work on subject more closely related to robotics. Finally my work can be splited in 3 main topics.[1]

First, I had to provide a design of a mechanical architecture with SolidWorks of the system they will then use to mount the LIDAR, the 3 stereo cameras and the 3 thermal cameras on top of the car.

Then, my tutor asked me to port a ROS package written in C++ to a ROS2 package in C++ too. The goal here was to be able to use a thermal camera with ROS2. Indeed, the constructor only provides a ROS1 package to use the thermal camera.

The third part of my work was to create a modular and reusable ROS2 package to make the calibration between the sensors. To accomplish this, I have designed several self-sufficient processes, each of which could be easily modified and adapted for other projects in the future.

[1] In this report, the different steps of my work are detailed one after another. However, during my internship, I worked on the different missions at the same time depending on the feedback I needed from my tutor to be more efficient.

The calibration package consisted of seven programs, including ones for saving images and point clouds, finding key points in images and point clouds, and obtaining extrinsic parameters using the RANSAC algorithm.

Finally, my tutor asked me to create an english report about my work to be able to reuse it in the future. This report is 15 pages long and explains precisely the few steps to download and use the packages, why the SolidWorks model was made this way etc.

Because of my substitution of the 4th semester at ENSTA Bretagne I didn't learn how to code with ROS and how to process images contrary to my classmates. This internship was the opportunity to take time working on it to be able to easily follow the lessons of my last year of study.

The research's center missions

My internship took place at the Escuela Técnica Superior de Ingeniería Industrial of Málaga, Spain.[2]. From the 10th of May to the 28th of July I joined the Robotics and Mechatronics Group. My tutor, Dr. Jesús Morales Rodríguez is a member of the and the Research one. The Figure 2 represents a diagram of the organization of the Robotics & mechatronics group.

The emergency team goal is to develop robots to help first responders in potentially hazardous places [2]. The research team works on different topic such as the PREMOVE project which aims to develop develop innovative technologies and infrastructures for efficient, effective and safe urban transport, with a focus on electric vehicles [9]. The Figure 3 is one of the robots the researchers are working one to help first responders.

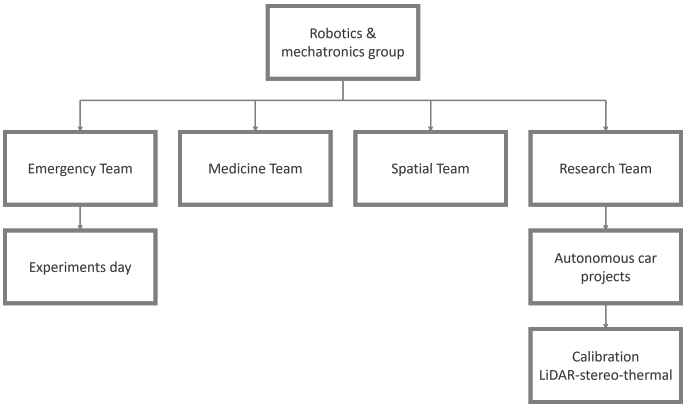


Figure 2: Teams diagram

Figure 3: Robot J8

I also had the chance to participate at the annual "Jornadas Internacionales de la Universidad de Málaga sobre Seguridad, Emergencias y Catástrofes" (JEMERG 2023) organized by the University of Málaga the 8th and 9th of June. This event is composed of a day of conferences on current topics related to Security, Emergencies, and Disasters the 8th of June. The 9th of June was a day of experiments to reproduce disaster scenes and allow rescue

workers to train in real-life conditions in cooperation.

The robotic Emergency Team have tested their robots in real conditions in collaborations with the firefighters, "la Legion" (a Spain Land Army unit), the paramedics, the Guardia Civil and some companies.

The mission of the J8 was to follow the soldiers of "la Legion" during their real condition training. The mission of the soldiers was to reach an area and simulate an ambush. During the exercise, one soldier act as if he was injured by an enemy. As the soldier needs to be rescued, the J8 have to reach the area of the ambush and when the soldier is lying on the robot, the J8 follows the rest of the soldiers in formation to go back to a safe area.

The "Follow me" mode allows the rescuers to be followed autonomously by the robot thanks to its LiDAR. It will adapt the current direction and speed. If the rescuer starts running, J8 is able to accelerate. On the contrary if the rescuer stops quickly, it is also able to decelerate in a second not to hit the people in front of it.

The Search & Rescue team develops inertial navigation and path finding algorithms to add a higher level of abstraction and make the robot fully autonomous. By using an elevation map given by a flying drone, the algorithm finds a quick and safe way to reach the mission. The map needs to be actualised frequently. Indeed, even if Málaga is mainly hot and dry, the region can face rain or wind some days of the year. In addition, the robot may be used in northern region of Spain for some missions in the future and needs to be robust to every situations.

1 Conception of Solidworks model

1.1 Context

During my internship I had to create a CAD model of the system to install everything and mount it on top of a NISSAN LEAF. To do that I used Solidworks 2023.

My work was to create a system which can be mounted directly on top of the car. The final system have 3 stereo cameras, 3 thermal cameras and a LiDAR. All of these sensors have different field of vue which need to be taken into account during the conception. I have created 6 versions before obtaining the final one. During the different steps, some aspects were tweaked to fix issues before ordering the real system to a manufacturing company. You can watch the full size figures of the different steps in the Annexe A.

1.2 My work

During my work, I tried to create a solid mechanical system which can be mounted thanks to the fixations of roof bars. It should be easily mountable and pieces should be easy to dismantle if the future researchers need to change the sensors.

The system was thought to be symmetric, easy to produce and modular. Every sensors have its own plate on which it is fixed. And every plate is fixed to a longer to unite them. The LiDAR need to be on top of every other sensor because it has a 360º FOV. If it was not on top, some plates would obstruct its vision which would make the system losing information about the environment. In addition, the vertical FOV of the LiDAR is not symmetric, it captures a 15º to the top and 55º to the down so it needs to be the mosts elevated sensor.

This first version shown in Figure 4 already had the wires of the sensors for the real system. The LiDAR has a metal piece which force the wire to go down so I had to create a hole in the plate to let it pass through. This first version was not the final version at all but allowed me to explain quickly my ideas. Then, I asked a first feedback to my tutor in order to modify it if he would expect something different.

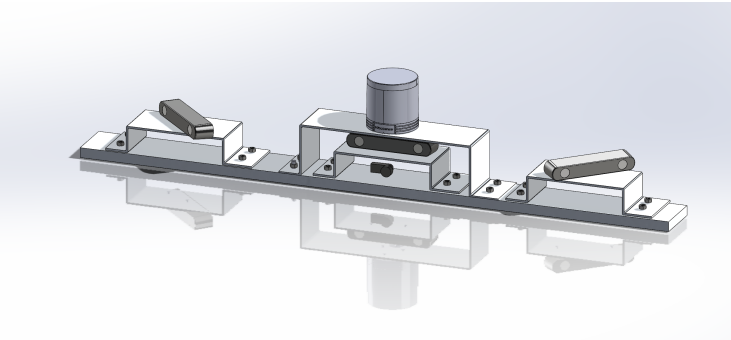


Figure 4: V1 of the SolidWorks model

My tutor liked my first ideas so I could go forward to a more precise version of the system. I removed the screws which made everything more complex each time I had to tweak something

because of the mates. I have created the little hole to fix the LiDAR and the 3 stereo cameras to their own plate because they already had tapping to fix them with screws. I also started to create a protection case for the thermal camera because it didn't have tapping to fix it. This case was changed multiple time in the next version because the thermal camera had lots of rounded shapes which made it hard to create a case without clearance. As you can see with Figure 5, at this moment, the case was fixed to the bottom part of the stereo plate and not to the big plate which supports every others. Indeed, the thinner the plate is, the tinier the screws are. And it is better for the case to have those big screws not to weaken it.

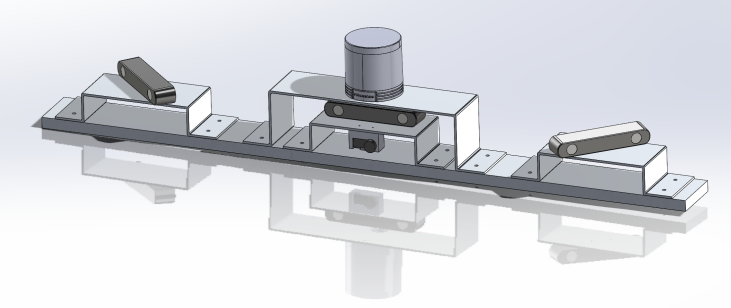


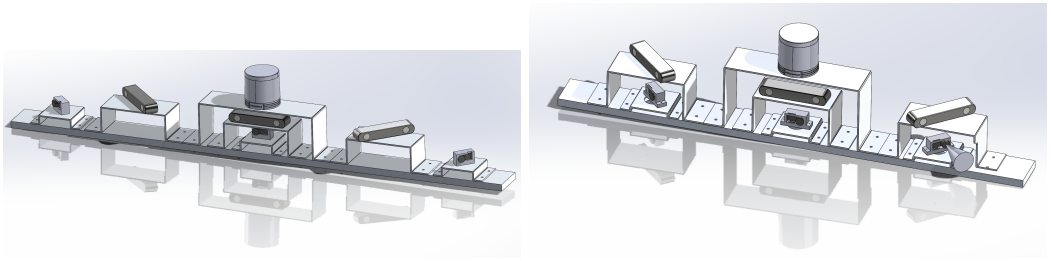
Figure 5: V2

In the third version, two other thermal cameras and their plates were added as you can see on Figure 6a. The main plate was lengthened by 30 cm which leads to a plate of 1m50, which is longer than the width of a NISSAN LEAF. After this part, my tutor and its colleague went to give me feedback on my work.

The version 4 that you can see in Figure 6b, have few modifications:

- Put the thermal camera under the stereo camera plates, which leaded me to cut a part of the right stereo camera plate not to obstruct the field of view of the thermal camera. I've added a cone to represent the FOV to be sure I didn't make any mistake with my on-paper calculus.
- Make the thermal camera pass through the bottom plate and not to the top for a better cable management. The camera would be upside down be it can be computed to correct it afterwards.
- Elevate the LiDAR to be sure it won't capture the front part of the car.

My tutor liked the design but wanted it be fully symmetric which was not the case anymore because of the thermal camera which is not symmetric.



(a) version 3 (b) version 4

Figure 6: versions 3 and 4

1.3 Results

In this final version, I completely modified the design of my thermal camera case to have a symmetric design were the center of the thermal camera is just under the center of the stereo camera above it. This implied to also cut the left stereo plate not to obstruct thermal camera FOV. I was also cautious about the stereo cameras FOV so I've reduced the lengths of their plates.

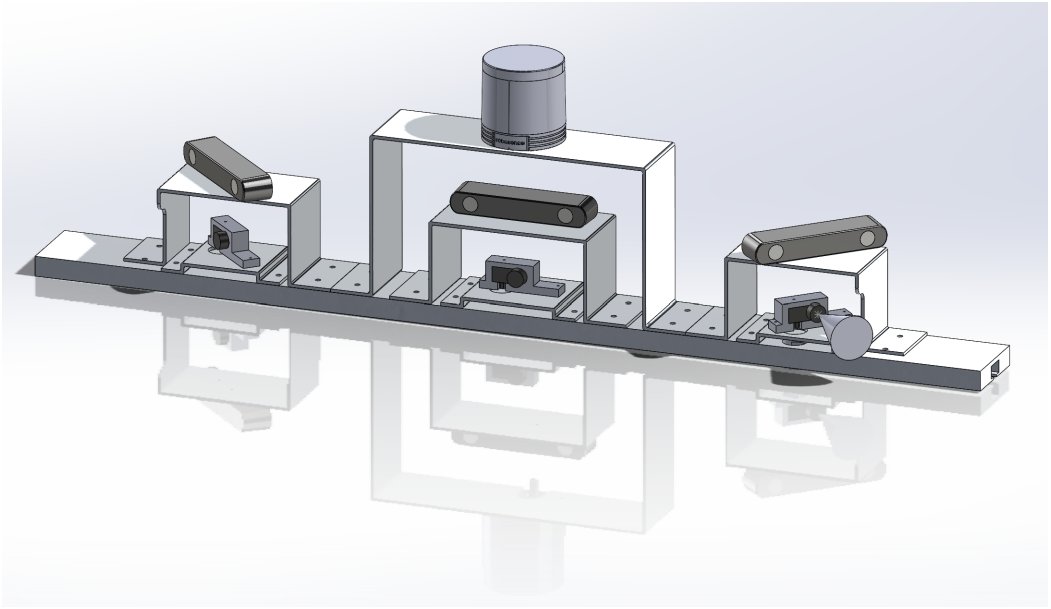


Figure 7: Final version of the SolidWorks model

The final version of the model is light and full of space for the wind to pass without receiving to much strength much can cause a loss in the aerodynamics. Every part in modular and can easily be replaced if wanted. Also this plate can be installed thanks to the roof bars fixations on top of the NISSAN LEAF. To present my work, I have created two SolidWorks Drawings with the different view of the system. These drawing are shown in Annexe A .

2 RO2 package porting

2.1 Context

In my internship I calibrated stereo cameras with LiDAR as explained in the next part. However, the full system also uses thermal cameras. The thermal cameras used are the Seek Thermal Compact Pro [13]. Which is an advanced and portable pocket-sized thermal camera. You can see what it looks like in Figure 8.

With its compact design, users can visualize and capture temperature differences with high thermal resolution. However the constructor give the package to use it with ROS but not with ROS2. Which is a problem as every other sensor can be used in ROS2 and my tutor wanted the calibration package I also had to create in ROS2. So my work was to port a package from ROS1 in C++ to ROS2 in C++ too.

Figure 8: SeekThermal Compact Pro

2.2 My work

The most difficult part of this package porting was to code the CmakeList.txt. In fact in the ROS1 package, a library with all the usefull functions to visualize the data with ROS was inside the main package. Moreover, in ROS1, namespace are commonly used to create objects in C++. These ways of coding the package was obsolete in ROS2 so I had to completely recode and face the problems it may cause. For example, lots of functions from ROS changed in ROS2 in their syntaxe and the way they are used.

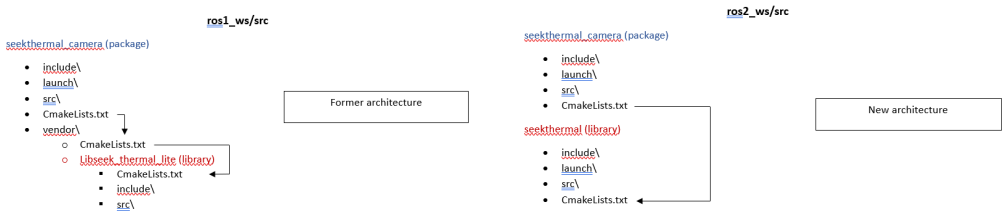
As explained in introduction, I didn't attend the ROS lesson of ENSTA Bretagne because of my semester substitution in Canada. So I had to learn ROS and ROS2 during my internship. My tutor gave me an account to learn ROS2 basics online on TheConstruct website [11] and I have completed my online formation thanks to forum, topics and websites such as ChatGPT. This website is a perfect to learn ROS2 with theory and exercises thank to the online simulator. The course was divided that way:

- Introduction to the Course
- ROS2 basic concepts
- Understanding ROS2 Topics
- Understanding ROS2 Services

- Executors and Callback Groups
- Understanding ROS2 Actions
- Debugging tools

This part of my internship was the most frustrating as creating a CmakeList with the architecture of ROS1 was impossible. Moreover, to use the functions from the ROS1 package, I had to use OpenCV3 but to work with the other sensors I had to install a version of CUDA which created lots of issues with this version of OpenCV. To install it properly, I had to uninstall lots of low-level library from the computer and sometimes, it created lots of issues with the computer. I asked my tutor to help me during the installation of CUDA and OpenCV3 and after one week, we were finally able to use them. I explained every step needed to download them properly in the user report my tutor asked me to write.

To make the package working in ROS2 I had to completely change the architecture of the package. In ROS1, the package was in the src file of my ros1_workspace and the library was inside this package. In this configuration, the CmakeList.txt of the package was directly connected to the CmakeList.txt of the library thanks to the command add_subdirectory(). However, it didn't work in ROS2 so I modified both the package and the library codes and then I put both of them in the src file of my ros2_workspace. The Figure 9 represent the difference between ROS1 and ROS2 architecture.



(a) ros1 architecture (b) ros2 architecture

Figure 9: Different architecture

2.3 Results

You can see in Figure 10 the results of my ROS2 package. In the first plan my left hand and in the second one my shirt. The background is darker than my body because of the heat the thermal camera detect.

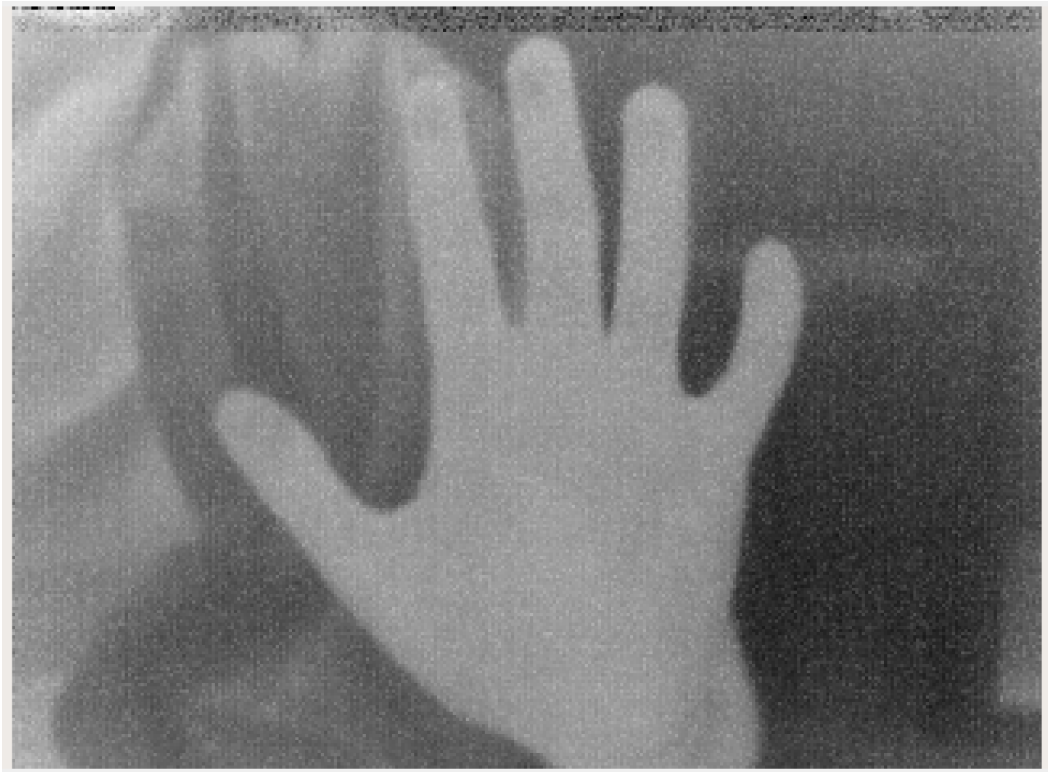


Figure 10: Visualization with rqt of the thermal camera vision

During the porting from ROS1 to ROS2, a darker line has appeared on top of the visualisation. This problem don't appear in the ROS1 package and the dead pixels are static. I didn't figure it out why it appears but as this tiny line is only on top of the visualisation and not in the center of it my tutor told me he was satisfied with this work so I could go to the next mission of my internship.

In the end, it is a better architecture for two reasons. First, it is easier to understand the dependency between the library and the package. Moreover, it allows the user to use the library by itself in other packages.

3 Multi-sensor calibration

3.1 Context

3.1.1 Camera calibration

Camera calibration determines the camera's characteristics, correcting for distortions and providing a mapping between the 3D world and 2D image coordinates.

By calibrating a camera, we can correct lens distortion and obtain a mapping between the camera's image plane and the real world. This calibration is essential for a wide range of computer vision tasks, such as 3D reconstruction, object tracking, augmented reality, and more, where accurate knowledge of the camera's imaging characteristics is crucial to ensure reliable results. When we want to calibrate multiple sensors, the calibration of a single sensor is called "intrinsic calibration".

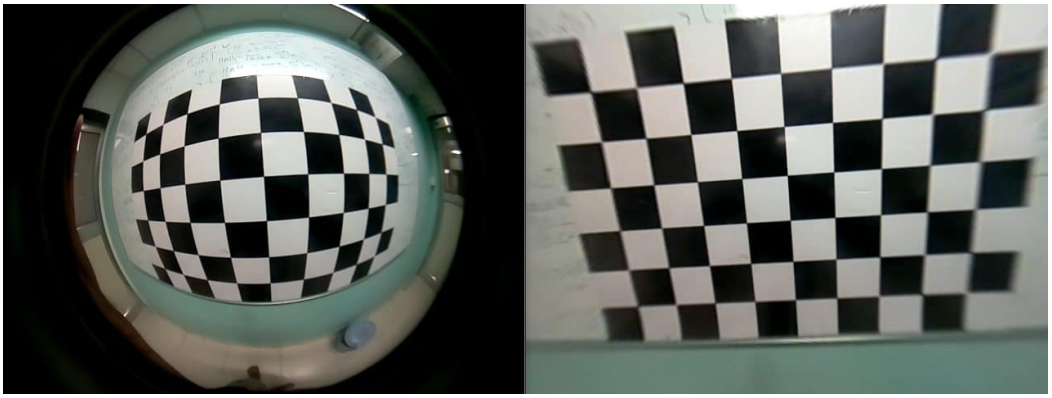


Figure 11: Camera calibration [14]

3.1.2 Multi-sensor calibration

Multi-sensor calibration aligns data from different sensors, like cameras and LiDAR, to establish accurate correspondences between their coordinate systems for improved accuracy and integration in various applications.

The calibration process involves finding the transformation parameters between each sensor's coordinate system and a common reference frame. These transformations enable the combination of data from different sensors, allowing the creation of a more comprehensive and accurate representation of the environment.

If finding the parameters of one camera to get rid of lens effects like distortion is called "intrinsic calibration". Finding the translation and rotation vectors between two sensors is called "extrinsic calibration". To be able to use information from multiple sensors to analyse an environment, we have to combine both of these parts, this process is called "Multi-sensor calibration". The Figure 12 is a representation of steps the user can follow to calibrate its system. You can also read the article [3] to see an example of a longer project on the subject of Multi-sensor calibration of RGB and thermal cameras with a LiDAR.

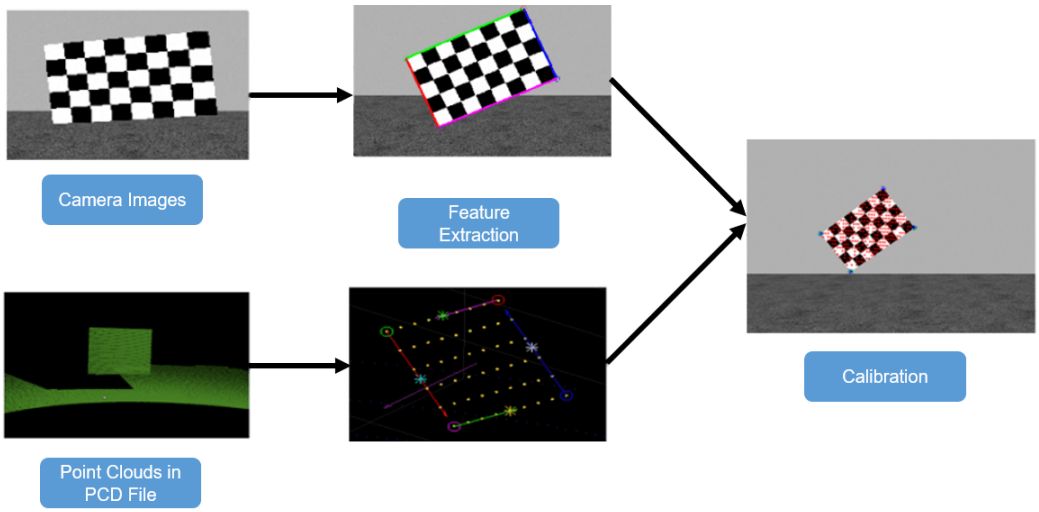


Figure 12: Camera calibration [5]

3.1.3 Mathematical approach

From a mathematical point of vue, the camera calibration is represented by a matrix multiplication. You can see it in the equation 1.

$$\begin{bmatrix} xc \\ yc \\ zc \\ 1 \end{bmatrix} = \begin{bmatrix} fx & 0 & cx & 0 \\ 0 & fy & cy & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} . \begin{bmatrix} r11 & r12 & r13 & tx \\ r21 & r22 & r23 & tx \\ r31 & r32 & r33 & tx \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} xw \\ yw \\ zw \\ 1 \end{bmatrix} \quad (1)$$

The intrinsic parameters are :

- fx and fy represent the focal length in the x and y axis. These two parameters may be different depending on the shape of the lens, in our case they were the same because the lens of the cameras are axis-symmetric.
- cx and cy that represent the coordinates of the optical center projection onto the image plane. For exemple if the camera is 1280x720, cx is close to 640px and cy is close to 360px.

On the other hand, the extrinsic parameters are:

- rij the parameters of the rotation matrix R3x3, the rotation matrix that transforms from the world coordinate system to the camera coordinate system.
- tx, ty and tz, the components of the translation vector that transform from the world coordinate system to the camera coordinate system.

xw,yw and zw represent the coordinates of the point in the real 3D-world. xc and yc represent the 2D-coordinates of the point in the frame of the camera. zc is not readily interpretable, it is only a variable which can helps the user to correct xc and yc if the sensor of the cells of the camera are not orthogonal.

3.1.4 LiDAR: Helios 5515 LiDAR from Robosense

The RoboSense Helios 5515 shown in Figure 13 is a high-performance LiDAR sensor designed for autonomous driving and robotics. With its advanced technology, long-range perception, and precise object detection, it delivers superior 3D point cloud data for safe and efficient navigation in complex environments. It's an ideal choice to power next-gen autonomous systems. You can compare it to the others Robosense LiDAR on the [10] And buy it on ROS components web page [1].

Figure 13: Robosense Helios 5515 LiDAR

3.1.5 Stereo Cameras: ZED2i stereo camera from Stereolabs

The stereo camera we used is the ZED2i stereo cameras [12]. The ZED2i camera is a stereoscopic camera used for 3D vision and mapping. It uses two image sensors and a stereo vision system to capture 3D images and videos. The ZED2i camera is compatible with multiple development platforms, including ROS2 and OpenCV, making it a popular choice among developers and researchers working in the fields of computer vision and robotics. Figure 14 shows a picture of this camera.

Figure 14: Stereolabs ZED2i stereo camera

3.2 My work

As I had few times to create a full multi-sensor calibration process, I wanted to separate the full process into multiple self-sufficient processes that can be modified easily. This way my work could be reuse in the future, adapted to other programs. To have a multi-sensor calibration done, the package has 6 python scripts:

- save_image.py and save_pointcloud.py: save images and point clouds,
- open_modify_image.py and open_modify_pointcloud.py: find the 2D key points in the images and the 3D key points in the point cloud,

- calibration.py Get the extrinsic parameters associated with this key points using RANSAC algorithm. Then visualize the results of your calibration.
- calib.py: library to store all the utility functions the other programs use.

All these steps can be followed by launching the different nodes of the package calibration_pkg.

3.2.1 Save images and point clouds

The steps are exactly the same for the image and the point cloud. So everything explained here for the pointcloud can be transposed for the image. First, the user needs to launch the program: save_images.py. Before launching the program, the user modifies the path of the folders where the images and the point clouds need to be saved on its computer. As the messages are taken from the ROS2 messages delivered by the sensors, the user also has to modify the names of topics in the creation of subscribers.

After this step, the user can launch the program and can click on the "V" key of its keyboard to visualize the point cloud. To be able to calibrate the system afterward, the user needs to be sure that it got an object with easy shapes like a cardboard. The user will then place the cardboard in different positions of the room and different inclination to get enough information. Obviously, the cardboard needs to be in the field of view of all the sensors used in the calibration, in our case, in the FOV of the LiDAR and in the FOV of the camera.

To simplify the following steps, the cardboard needs to be in a zone without any objects and with a uniform background. If it is done, the user will easily make the distinction between the background and the cardboard. Finding the corners will be easier. Similarly, the color of the background needs to be different from the one of the carboard, otherwise, it will be hard to find the exact pixel of corners in the image.

When all these steps are checked, the user can press the "S" key to save the point cloud. This first step is now finished.

Figure 15: Cardboard in top left corner

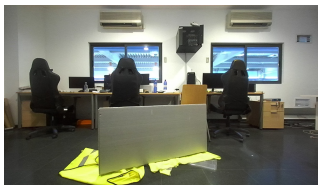


Figure 16: Cardboard in bottom center

Figure 17: Cardboard in top left center

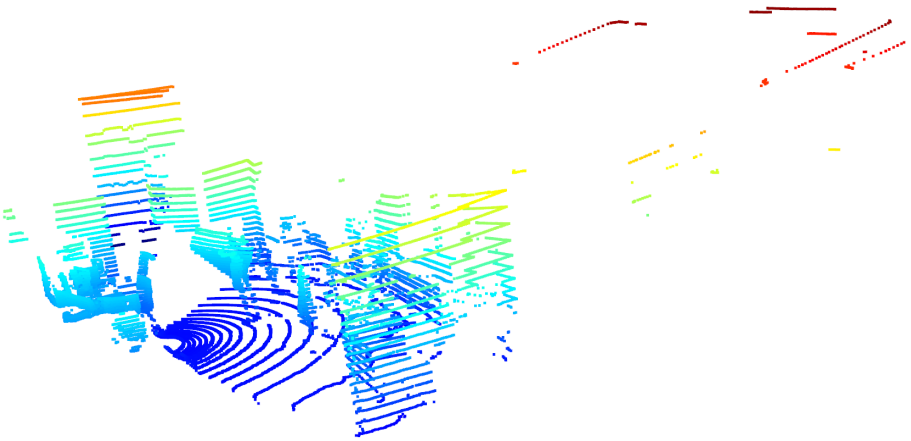


Figure 18: View of a pointcloud

3.2.2 Find key points in images and point clouds

This step allows the user to identify the coordinates of the corners of the cardboard in the frame of the image and the point cloud. Each point studied needs to be the exact same point in point cloud and the image. That's why we focus on the corners of the cardboard and we want a simplified environment with uniform background. The user needs to modify the path of the images it has saved. Then it can launch the visualisation by clicking on the "V" key. By clicking on the "P" key, it can print the coordinates of all the points displayed. If there are too much points, the user can modify the selection zone to find more precise coordinates. The keyboard is a Spanish one, so to be more ergonomic, the user needs to change the keys in the program if it uses a French or American one for example. In Figure 19 you can see the cardboard diagonally posed with all the 3D points. In Figure 20, you can see the same pose for the cardboard but with less 3D points around, which makes it easier for the user to find points at the corners.

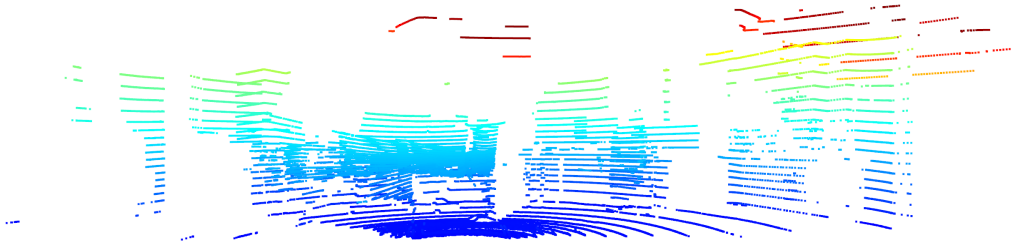


Figure 19: Full point cloud

Figure 20: Only the cardboard

3.2.3 Get extrinsic parameters and display

The user now needs to store the coordinates of the 3D and 2D points it finds in two lists, respectively: objectPoints and imagePoints.

To find the extrinsic parameters, the program is based on the solvePnPRANSAC Algorithm. SolvePnPRANSAC stands for "Perspective-n-Point RANSAC," where PnP refers to the perspective-n-point problem, and RANSAC stands for Random Sample Consensus, a robust estimation algorithm. Combining the PnP problem with RANSAC, solvePnPRANSAC robustly estimates the pose of an object by iteratively sampling subsets of data, fitting a model (pose) to each subset, and selecting the best-fitting model with minimal influence from outliers.

The SolvePnPRANSAC algorithm needs at least 4 points. I advice to the user to find at least 4 points coordinates of the cardboard (the corners for example) in the same image/point cloud. And to study at least 3 different poses of the cardboard to find correct extrinsic parameters. The more points the algorithm has, the more precise the extrinsic parameters will be.

The algorithm also need the intrinsic calibration matrix of the camera. In our case, these parameters are given by tone of the topics of the camera in real time. We will use the center of the image (cx, cy) and the focal distance in the x axis and the y axis: (fx, fy). When we have all these intrinsic parameters, the function from calib.py compute the intrinsic matrix.

When the program is launched, it will compute the rotation vector rvec and the translation vector tvec and print them to the user.

Finally, the program takes the rotation and translation vectors found before and subscribe to the LiDAR and camera topics to display in real time the information fused thanks to the projection of LiDAR points on the 2D image.

3.3 Results

Now that the user has followed all the steps explained in previous part, it can visualize the multi-sensor calibration it got. If it is not satisfied of the result, it can create new samples to improve the extrinsic parameters.

However, with all these steps done, the visualisation is a bit chaotic as you can see in figure 21. Indeed, from a computational point of vue, everything is correct, but we now need to clear the data to visualize the calibration of objects.

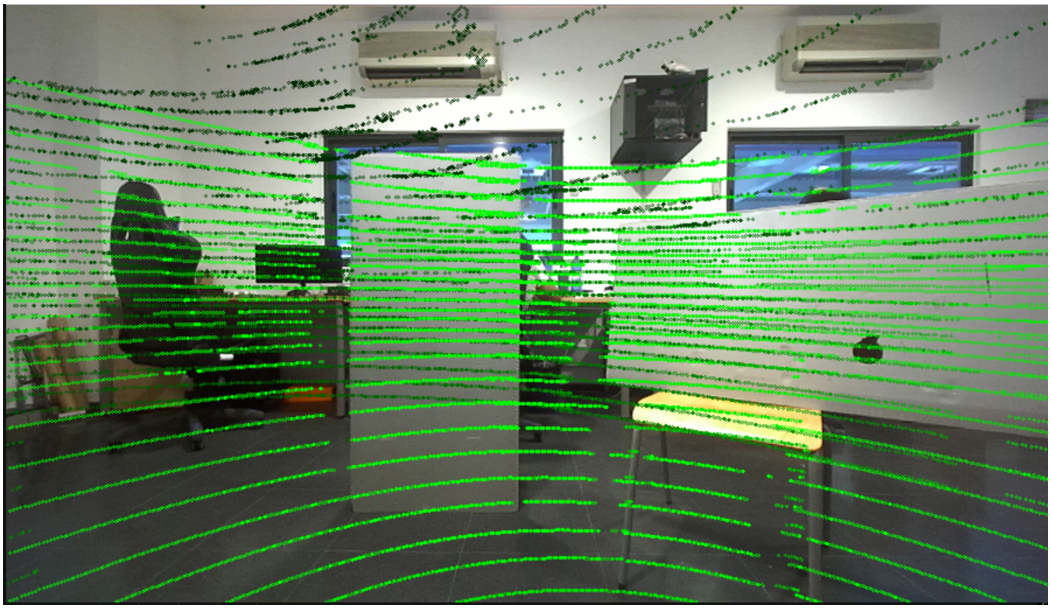


Figure 21: All points projected

3.3.1 Different zones

First we need to remove the 3D points from the LiDAR point could which represents objects behind the stereo camera. Indeed, during the projection phase, every 3D points of the space is projected in the 2D plane of the camera, no matter if they were behind or in front of the stereo camera. I have made a illustration in Figure 22 to represent the issue. The green points are the one in the FOV in front of the stereo camera. The other points are not in the FOV so they should not be projected in the image plane. For example, the grey points will never projected to the image 2D plane because they are not in the green or red corner. However, the points in the red corner are displayed even if they are not supposed to be. Indeed the computation of their 3D coordinates multiplied by the matrices give them 2D coordinates in the frame of the camera.

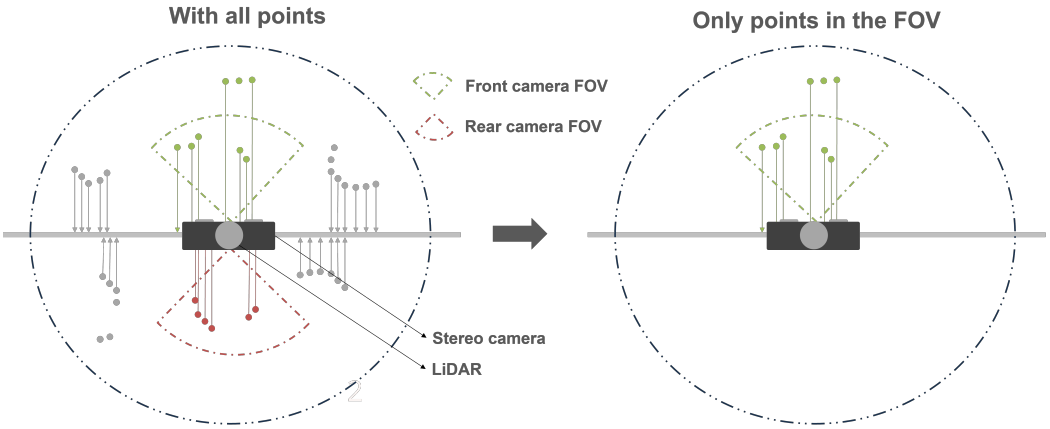


Figure 22: Correction of the projection

The naive way of erasing these points could be to only display the points that have negative x coordinates. This way, red points and grey points behind the camera are not displayed and the grey in front of the camera are not displayed too because they are not in the FOV, they are so far from the center that they are not displayed. But this way of doing it make the code save all those useless points are slow down the process dramatically. It creates too much latency and the user isn't able too use the program.

A better way of doing this step is not to store the point which don't satisfy Equation 2, with xw,yw,zw the coordinates of the 3D-points in our world frame and FOV, the field of view of the stereo cameras.

$$\begin{cases} -\frac{FOV}{2} < arctan(\frac{y}{x}) < \frac{FOV}{2} \\ -\frac{FOV}{2} < arctan(\frac{z}{x}) < \frac{FOV}{2} \end{cases} \tag{2}$$

The points that don't satisfy this equation are not saved in the list so they are not treated in the code. This way of doing allows the user to get rid of former latencies.

With this first correction we get rid of points that should not be displayed. But we can do better if we want to only see on particular object for example. The second step is to reduce the studied zone to focus on one particular object. It is only to see the information better but is not necessary to implement it because from a calibration point of vue, every point displayed should be displayed. It is only to visualize if our previous steps are correct. For example, in the Figure 23, we didn't wanted the background points of Figure 21 so we can focus on closer points as in Figure 24a. It is now easy to look at the calibration of the cardboard. We can also reduce the box to also To modify that we can use the same idea as when we have found extrinsic parameters : adding thresholds for the visualization.

This way, every point which is too far away can be erased to focus on one particular object. The Figure 23 is a drawing that represents that situation. The user can focus on one particular box of the world represented by darker grey points.

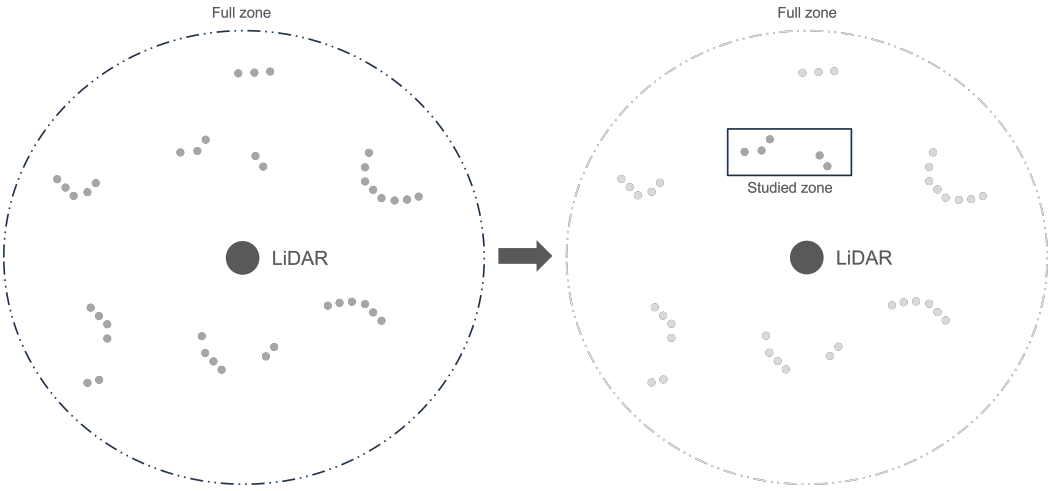
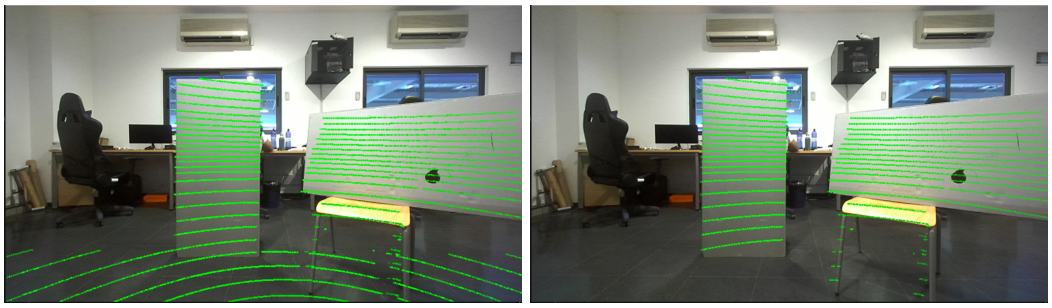


Figure 23: Reduced zone representation

You can see the difference between each correction step by comparing Figures 21 and 24.



(a) Without behind and floor points (b) Best display zone

Figure 24: Different extrinsic parameters

3.3.2 Different extrinsic parameters

Finally, the user can observe correctly the results of its calibration. To visualize the importance of finding extrinsic parameters thanks to lots of samples, Figure 25 represents an example of a bad and a better calibration. The only difference between both is the number of keypoints found. The more you have, the better it is for the calibration. It is also better if the user put its object in different positions and rotations of the camera FOV. For example, Figure 25a is obtained by taking only 3 points of 2 different positions of the object.

On the contrary, Figure 25b is better, it has been realised with 4 points in 4 different positions. Of course, if the calibration need to be implemented for real conditions and not for experiments, the user need to take way more different positions of the object to improve the accuracy. Specialists of multi-sensor calibration consider that around 10 different poses are necessary to have a very accurate calibration. [3]

(a) Bad calibration (b) Better calibration

Figure 25: Different extrinsic parameters

I am satisfied with my work here. The packages created are easy to use and really modular so it will be easy for other searchers to modify a part with completely change the rest of the code. For instance, the part of the process to determine the coordinates of corners in 2D and 3D can be modified and it won't impact the part to save the pointclouds and images.
However, if I had more time, I would have liked to speed up the process to find extrinsic parameters. In fact, to have a good calibration, the user need to give lots of 2D-keypoints and their corresponding 3D-keypoints coordinates to the RANSAC algorithm to find accurate extrinsic parameters.

The process to find these 2D and 3D coordinates could be automatized like in the work of Michael May [6]. To do it, the best way is to use OpenCV functions to detect the keypoints of the camera image with an object like a chessboard. If the user know exactly the shape of this cardboard (number of cells and their size), OpenCV algorithm are able to detect the keypoints at the intersection of the black and white cells.

A similar process can be done in 3D-pointcloud of a LiDAR with Open3D, this time the functions will detect the edges of the object, so the center point of this object and the normal of the plane passing through this point. OpenCV or Open3D functions allows to find the extrinsic parameters [4]. This way of doing it is quicker and automatic and I would have liked to add it, however I could not because of a lack of time. My process is working as I wanted and can easily be improved by next works as it is very modular.

4 Conclusion

In conclusion, my internship at the Escuela Técnica Superior de Ingeniería Industrial in Málaga, Spain, was an enriching and rewarding experience, as I had the opportunity to work with the Robotics and Mechatronics Group on their missions to develop innovative robotic systems. The center's objectives of creating robots to assist first responders in hazardous situations and advancing urban transportation through electric vehicles aligned perfectly with my passion for robotics.

During my internship, I was primarily focused on multi-sensor calibration, a critical aspect of robotics development that aligns data from different sensors to ensure accurate integration and improved accuracy in various applications. I learned about camera calibration, which involved correcting lens distortions and mapping 3D world coordinates to 2D image coordinates.

In my work, I designed a modular and reusable multi-sensor calibration process, which included several self-sufficient programs. These programs allowed for the saving of images and point clouds, finding key points in images and point clouds, and obtaining extrinsic parameters using the RANSAC algorithm. The success of this work was demonstrated through various experiments, highlighting the importance of precise calibration for optimal robotic performance.

Additionally, I had the opportunity to observe the Robotics and Mechatronics Group's experiments during the "Jornadas Internacionales de la Universidad de Málaga sobre Seguridad, Emergencias y Catástrofes" (JEMERG 2023) event. Witnessing the robotic systems being tested in real-life scenarios in collaboration with first responders and emergency teams was a wonderful experience that showcased the practical applications and impact of the research.

I'm grateful for the project I worked on because it allowed me to learn ROS2 and image processing. These skills were missing from my education due to my time in Canada, and I would have needed them in my final year of robotics studies. Now, I feel confident using these tools for future projects.

If I had more time, I would have liked to mount the mechanical system I have designed with SolidWorks to see my work in real conditions. However, the mechanical parts were not ordered from a supplier before my internship ended.

Additionally, I would have liked to perform the calibration of thermal cameras. The process is similar to that of RGB cameras since, from a ROS perspective, the information retrieved consists of images received by subscribers in both cases. I believe that with an additional three weeks, I could have successfully implemented it. The thermal images have low resolution, which complicates the task. Furthermore, unlike stereo cameras, the focal lengths and distortion coefficients were not provided by the manufacturer, so I would have had to address this step before proceeding with the external calibration. While for RGB cameras, a simple black and white checkerboard pattern is enough for calibration, but creating contrast between the background and foreground in the case of thermal cameras would have required heating a metal checkerboard-shaped object. I have detailed my research and thoughts on this matter in the

report submitted to the university, aiming to provide a foundation for future work on this project.

Lastly, I would have liked to modify my approach to finding extrinsic parameters to automate the process, as explained in Part 3. By automating the process, the duration of this step could be drastically reduced, which is significant when aiming for reliable calibration across numerous samples.

My internship at the Escuela Técnica Superior de Ingeniería Industrial was a rewarding experience, allowing me to contribute to cutting-edge research and gain expertise in multi-sensor calibration and ROS2. The work conducted during this internship has further enhanced my passion for robotics, inspiring me to continue exploring innovative solutions for the advancement of technology in the field of robotics.

5 List of figures

List of Figures

6 Bibliography

References

[1] ROS components. *Buy Helios 5515*. URL: https://www.roscomponents.com/en/lidar-laser-scanner/312-rs-helios-5515-.html. (accessed: august 2023).

[2] *Emergency Team*. URL: https://www.uma.es/robotics-and-mechatronics/info/107721/FIRST-ROB/. (accessed: august 2023).

[3] Aravindhan K Krishnan and Srikanth Saripalli. *Cross-Calibration of RGB and Thermal Cameras with a LIDAR for RGB-Depth-Thermal Mapping*. URL: https://www.worldscientific.com/doi/epdf/10.1142/S2301385017500054. (accessed: august 2023).

[4] GAjay Kumar et al. *LiDAR and Camera Fusion Approach for Object Distance Estimation in Self-Driving Vehicles*. URL: https://www.researchgate.net/publication/339505716_LiDAR_and_Camera_Fusion_Approach_for_Object_Distance_Estimation_in_Self-Driving_Vehicles. (accessed: august 2023).

[5] Matlab. URL: https://es.mathworks.com/. (accessed: august 2023).

[6] Michael May. *Lidar Camera Calibration*. URL: https://www.youtube.com/watch?v=x5HLYL_B65g. (accessed: august 2023).

[7] *PREMOVE paper*. URL: https://www.uma.es/robotics-and-mechatronics/info/138109/premove/. (accessed: august 2023).

[8] *PREMOVE project*. URL: https://www.uma.es/robotics-and-mechatronics/info/138109/premove/. (accessed: august 2023).

[9] *Research Team*. URL: https://www.uma.es/robotics-and-mechatronics/info/88243/smart-cities/. (accessed: august 2023).

[10] LTD) RoboSense (Suteng Innovation Technology Co. *RS-Helios*. URL: https://www.robosense.ai/en/rslidar/RS-Helios. (accessed: 07.12.2023).

[11] *ROS2 basic courses*. URL: https://app.theconstructsim.com/courses/61/. (accessed: august 2023).

[12] StereoLabs. *Zed2i*. URL: https://www.stereolabs.com/zed-2i/. (accessed: july 2023).

[13] Seek thermal. *Compact Pro*. URL: https://www.thermal.com/compact-series.html. (accessed: july 2023).

[14] Intel et Willow Garage. URL: https://learnopencv.com/camera-calibration-using-opencv/. (accessed: august 2023).

A 3D models of the system

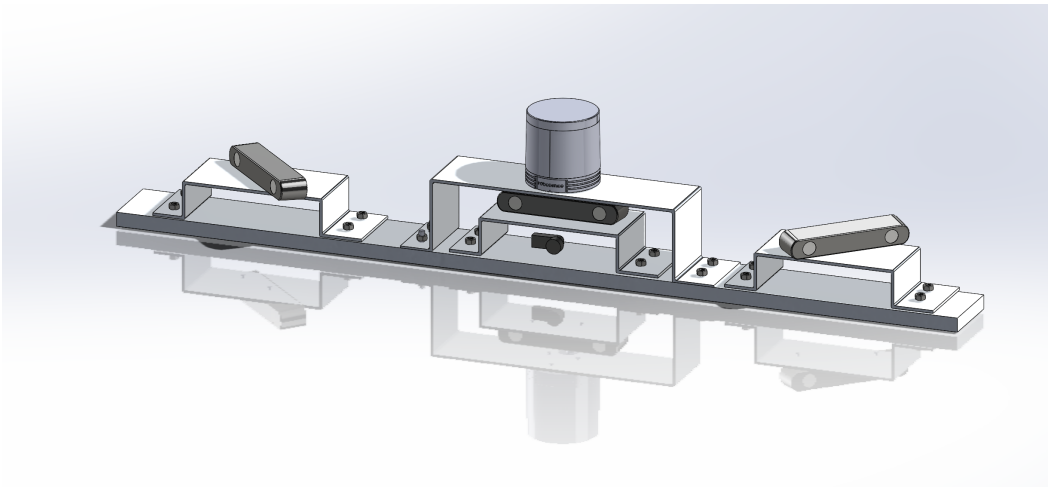


Figure 26: V1

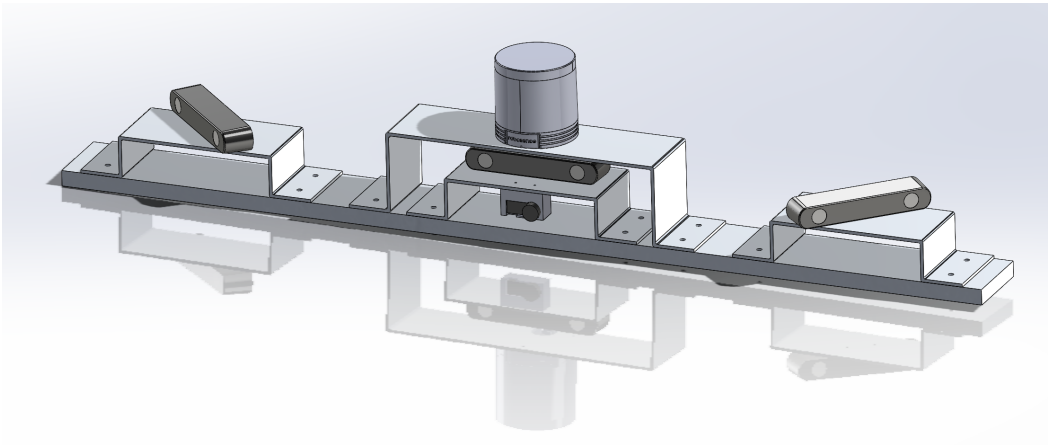


Figure 27: V2

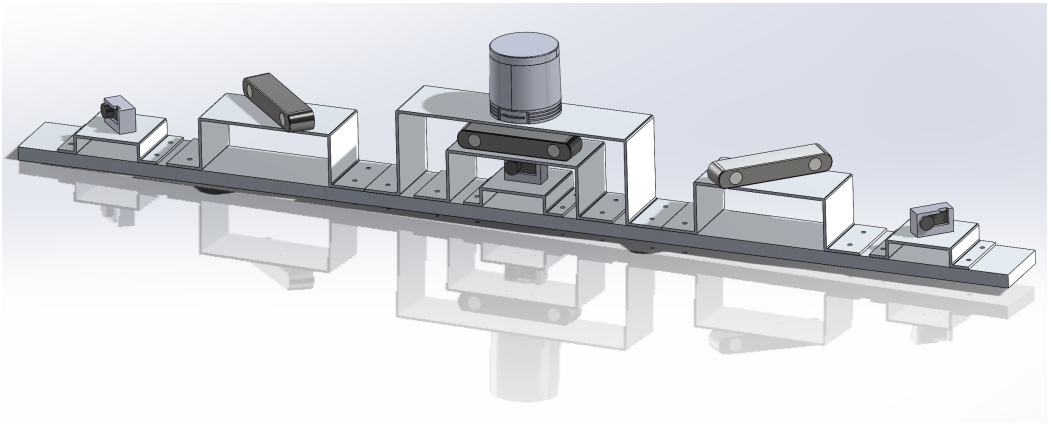


Figure 28: V3

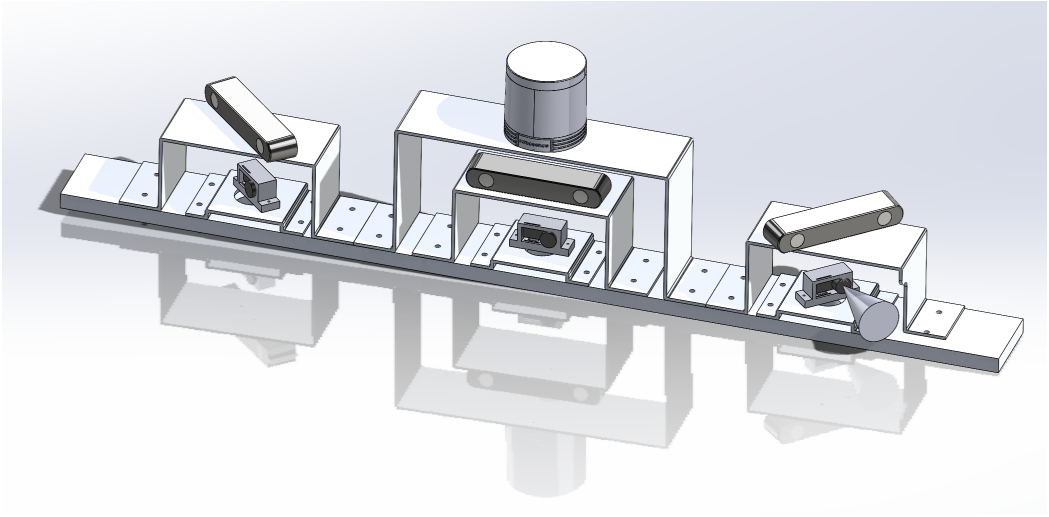


Figure 29: V4

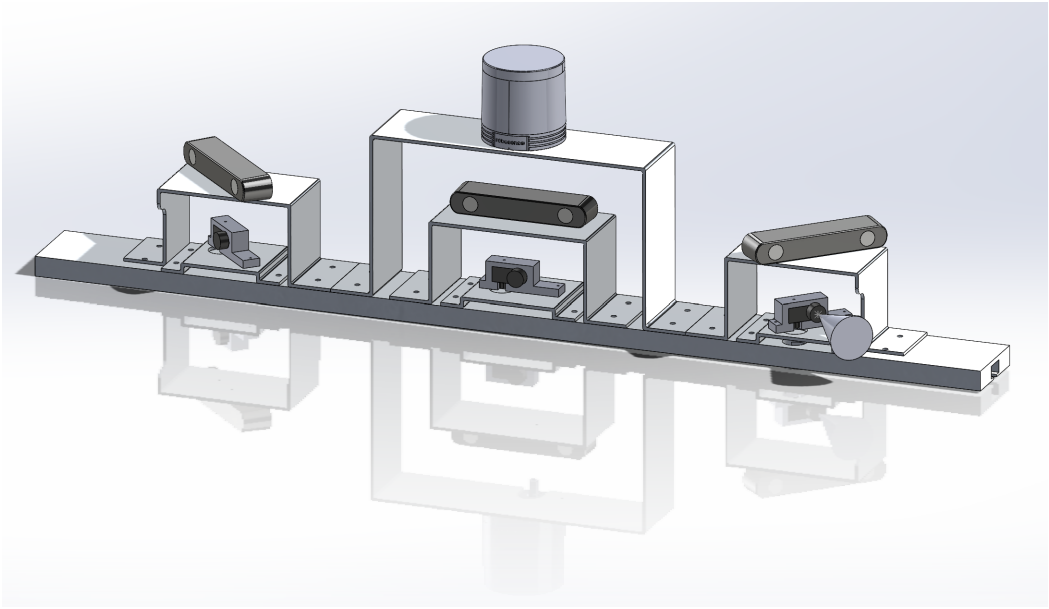


Figure 30: Final version of the SolidWorks model

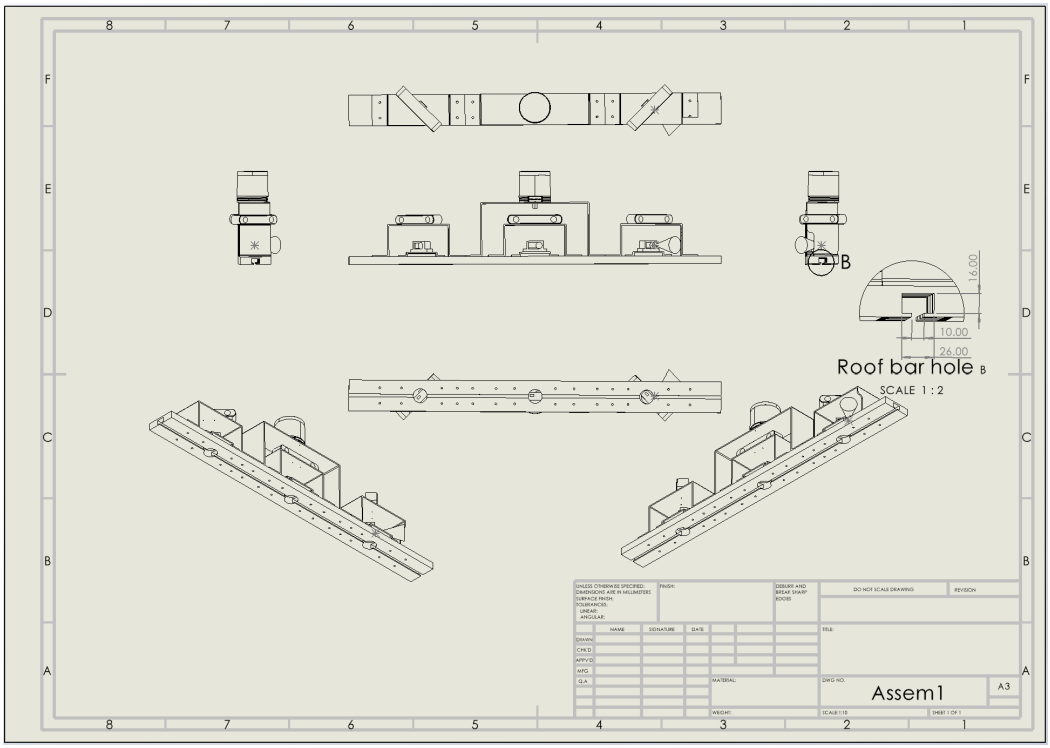


Figure 31: Drawing of the full system

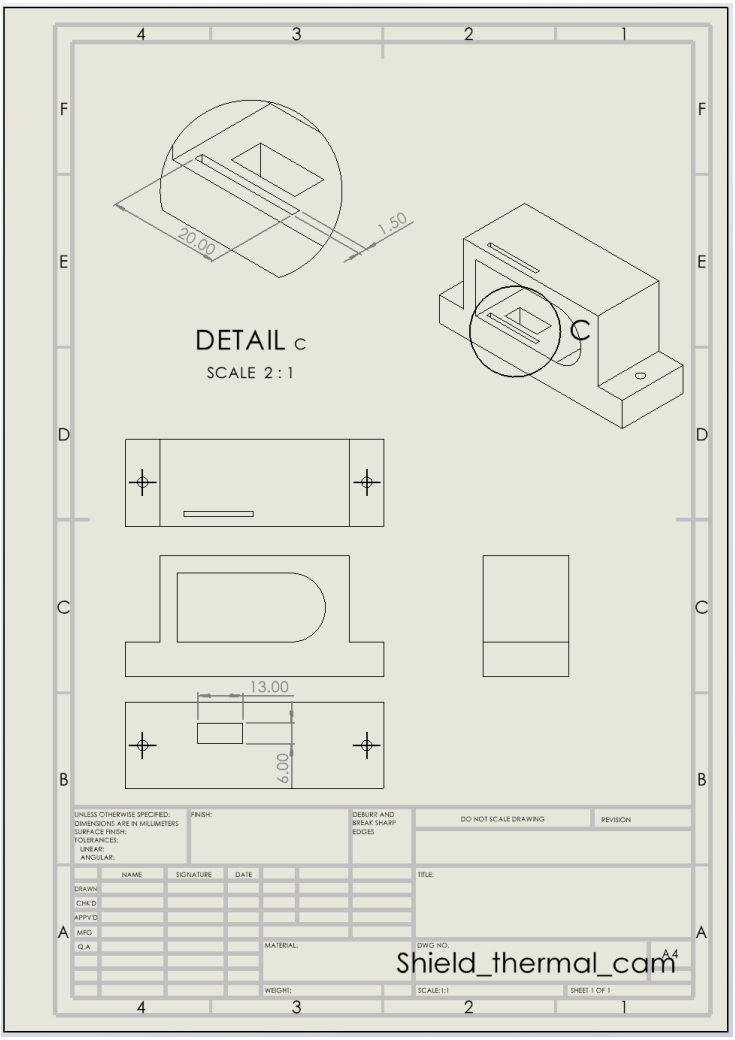


Figure 32: Drawing of the protection case

Preliminary work on multisensor calibration in the PREMOVE project

BELIER Titouan[1]

[1]**titouan.belier@ensta-bretagne.org**

ABSTRACT

During a 3 months ERASMUS internship in the context of the PREMOVE project [8] at the Escuela de Ingenierias Industriales under the supervision of Dr. Jesus Morales Rodriguez. The goal of this internship was to develop a procedure to calibrate a multi-sensors system integrated by 3D LiDAR, 3 stereo cameras and 3 thermal cameras. We will first explain what components we have chosen, how to install their ROS2 packages and how they work. Then, we will study the SolidWorks model created to attach the system on top of an autonomous car. Finally, we will explain how to calibrate the cameras to them and show the results.

Keywords: Multi-sensors calibration, ROS2, LiDAR, Stereo camera, Thermal camera, ROS2 porting, SolidWorks, PREMOVE

CONTENTS

1 HARDWARE SETUP AND PC

1.1 LiDAR: Helios 5515 LiDAR from Robosense

The RoboSense Helios 5515 shown in Figure 1 is a high-performance LiDAR sensor designed for autonomous driving and robotics. With its advanced technology, long-range perception, and precise object detection, it delivers superior 3D point cloud data for safe and efficient navigation in complex environments. It's an ideal choice to power next-gen autonomous systems. You can compare it to the others Robosense LiDAR on the [9] And buy it on ROS components web page [2].

Figure 1. Robosense Helios 5515 LiDAR

1.2 Stereo Cameras: ZED2i stereo camera from Stereolabs

The stereo camera we used is the ZED2i stereo cameras [13]. The ZED2i camera is a stereoscopic camera used for 3D vision and mapping. It utilizes two image sensors and a stereo vision system to capture 3D images and videos. The ZED2i camera is compatible with multiple development platforms, including ROS2 and OpenCV, making it a popular choice among developers and researchers working in the fields of computer vision and robotics. Figure **??** shows is a picture of this camera.

Figure 2. Stereolabs ZED2i stereo camera

1.3 Thermal Camera: Seekthermal Compact Pro

The Seek Thermal Compact Pro [17] is an advanced and portable pocket-sized thermal camera. You can see what it looks like in Figure 3. With its compact design and smartphone connectivity, users can visualize and capture temperature differences with high thermal resolution. Ideal for use in various fields such as inspection, maintenance, hunting, and security, the Seek Thermal Compact Pro provides a high-quality thermal experience in a convenient and user-friendly format.

Figure 3. Seek Thermal Compact Pro thermal camera

2 INSTALLATION

Two packages were created during this work, you can find them on Github:

- A package to interface with Seek thermal compact pro camera with ROS2 [19],
- a package which allows users to calibrate LiDAR and thermal or RGB cameras [18].

2.1 ROS2 & Ubuntu

To use this work, you need to use ROS2 as every package was made for ROS2 Humble under Ubuntu 22.04. To install ROS2 Humble, you can follow the steps of the official ROS website: [10]

2.2 LiDAR package

The package we will download is named 'rslidar_sdk'. First, download the package from the official GitHub of the package: [**rslidar**]. This package needs extra steps to work properly:

- Follow the steps to install the package in <ros2_workspace,>/src folder as written on the package GitHub. You will also need to install Yaml and libpcap,
- if you use ROS2, you have to modify the file package.xml and replace its content by the contents package_ros2.xml,
- to specify the type of LiDAR you are using, modify the file config/config.yaml by changing the lidar type to the one corresponding to your LiDAR. For example, in our case, the LiDAR type is RSHELIOS.

2.3 Stereo Camera package

To use our stereo camera we will need the zed _package [15]. To install it, it's pretty straight forward as for ROS2 installation. You will need to install two dependencies and the package:

- CUDA: [4],
- and ZED SDK: [14],
- and follow the steps of the GitHub.

After that, you only have to plug your ZED2i camera into a 3.2 USB port. Your camera is set up.

2.4 Thermal Camera package

This package is the most difficult to install properly. The package was originally made for ROS1 so we will divide this section in two parts. The first part is about installing the ROS1 package and in the 'How to use components' you will find how to visualize your thermal images in ROS1. The next part will be about installing a the package in ROS2.

2.4.1 Installing the ROS1 package

- Install Open CV3 by entering the first line of command of this file, [16].
- download libusb with this command line:

```
sudo apt install libusb-1.0-0-dev
```

- to download the ROS1 package you only have to follow that procedure written in this github page: [12].

2.4.2 Install ROS2 package

- Install Open CV3 by entering the first line of command of this file [16],
- then, to download the ROS2 package you only have to go to this link, download the library seek_thermal and the package seekthermal_camera,
- you first need to build the 'seek_thermal' library by executing the following commands:

```
cd path/to/seek_thermal/build
cmake ..
make
sudo make install
```

- yhen, you need to build your library with ROS2 to have your headers at the right place:

```
cd ~/ros2_ws
source /opt/ros/<your_ros2_distro>/setup.bash
colcon build --packages-select seek_thermal
source install/setup.bash
```

- to run the driver as an unprivileged user, set permissions for the camera device by copying the config/99-seekthermal.rules file from the repository to /etc/udev/rules.d and running the following commands: (the user accessing the camera must be a member of video group):

```
sudo udevadm control --reload-rules
sudo udevadm trigger
```

- Open the seekthermal_camera package, navigate to the CmakeLists.txt, and modify the path of the 'include' and 'lib' folder of your 'seek_thermal' library.

2.4.3 How to correct installation issues:

I can't build OpenCV3

If you have installed the stereo camera package, you have also installed CUDA. This software layer can cause you some troubles with OpenCV3 installation. To prevent this, add **-DWITH_CUDA = OFF** when you are building your package. OpenCV3 will now be installed without using CUDA but it's not mandatory and your package will work without it.

libjasper-dev is not found on my computer

When installing OpenCV3 you may encounter issues with the library 'libjasper-dev' library. If it's the case, use this link to find solutions [1]

Even with this change I can't build it

We encountered another problem when building OpenCV3—the 'make' command crashed during the process because we used too many threads. The computer we used had 24 threads, we had to only use 4 threads to make it compile properly: **make -j4**

I have a Cmakelist.txt problem

You may also have a problem when compiling your seekthermal_camera package because of your CMakelist.txt. To prevent it, navigate to your package >vendor >libseekthermal_lite >Cmakelist.txt. Open this file and replace the line find_packageOpenCV 3 REQUIRED COMPONENTS by **find_packageOpenCV 3 REQUIRED**.

3 HOW TO USE COMPONENTS

3.1 LiDAR

To launch the node of the LiDAR, you just need to open a terminal, source your ROS2 distro and use the launch command:

```
source /opt/ros/<your_ros2_distro>/setup.bash
ros2 launch rslidar_sdk start.py
```

The node launch a rviz2 window by itself but you can also do it yourself:

```
rviz2
```

3.2 Stereo Camera

To launch the node of the Stereo camera, you just need to open a terminal, source your ROS2 distro and use the launch command:

```
source /opt/ros/<your_ros2_distro>/setup.bash
ros2 launch zed_wrapper zed2i.launch.py
```

To visualize it you can use rviz2:

```
rviz2
```

3.3 Thermal Camera

3.3.1 Use ROS1 package

To launch the node of the thermal camera for ROS1, you just need to open a terminal, source your ROS1 distro and use the launch command:

```
source /opt/ros/<your_ros1_distro>/setup.bash
roslaunch seekthermal_camera seekthermal_pro.launch
```

To visualize it you can use rqt:

```
rqt
```

You will have this kind of image:

3.3.2 Use ROS2 package

To launch the node of the thermal camera for ROS2, you just need to open a terminal, source your ROS2 distro and use the launch command:

```
source /opt/ros/<your_ros2_distro>/setup.bash
ros2 launch seekthermal_camera seekthermal_camera.launch.py
```

To visualize it you can use rviz2:

```
rviz2
```

You will get resulting images such as Figure 4.

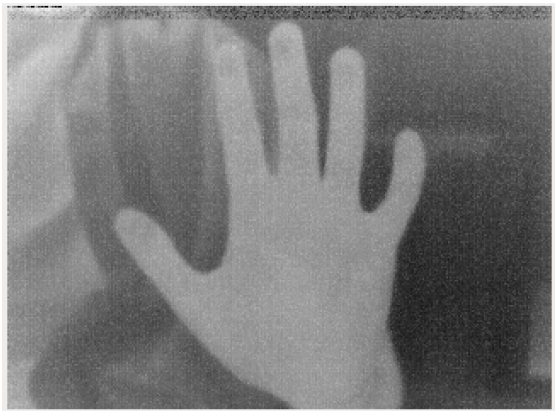


Figure 4. Thermal image in ROS2

4 SOLIDWORKS MODEL

The work done during the internship also contains a SolidWorks model of the final system with all the cameras and the LiDAR. The goal was to create a lightweight structure which will be attached to the roof bars of a Nissan Leaf car. The model is symmetrical from the point of view of the cameras. You can see the result in image Figure 5.

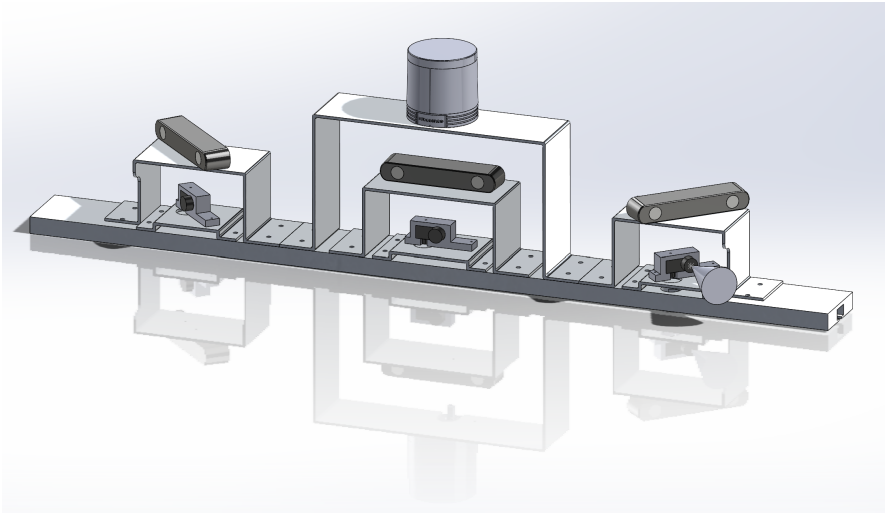


Figure 5. Final version of the model

The system is in three floors:

- The floor of the thermal cameras and their own 3D-printed cases,
- the floor of the stereo cameras,
- the floor of the LiDAR in the center of the system, it's on top to have a field of view which is not obstructed by other components.

The cameras and LiDAR can be bought in the websites mentioned in the introduction of this report. The thermal cameras cases need to be 3D-printed. All the other pieces can be bought from companies according to the SolidWorks model. The plates can be built in aluminium as it is lightweight and strong. The system is modular, each component can be added/changed/removed without impacting other components. The easiest way to install every components of the system is explained just above.

4.1 Installation guide

The system needs to be mounted like Russian nesting dolls, from the interior to the exterior:

First, put the thermal cameras in their 3D-printed cases as in Figure **??**. It is advised to use bolts and not screws to attach the 3D-printed camera to the plate as the plastic tapping can be damaged. The case could then move a bit and deteriorate the calibration. A little screw or pin can be added on top of the case to create another field of pressure on the thermal camera to be sure it won´t move.

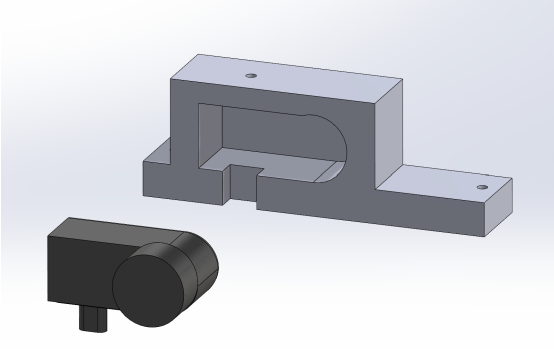


Figure 6. Step 1: install thermal cameras in their case

Then, attach these cases to their plates as in Figure **??**, take care that the wire of the thermal cameras goes down in the hole of their plates:

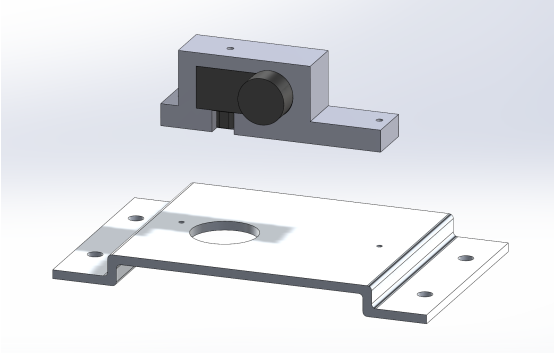


Figure 7. Step 2: attach cases to their plates

Now, do the same with the stereo cameras Figure **??**and the LiDAR Figure **??**. The wires of the stereo cameras needs to go at the back of the system and LiDAR cameras need to go down. There is a cut in its plate to pass the wire.

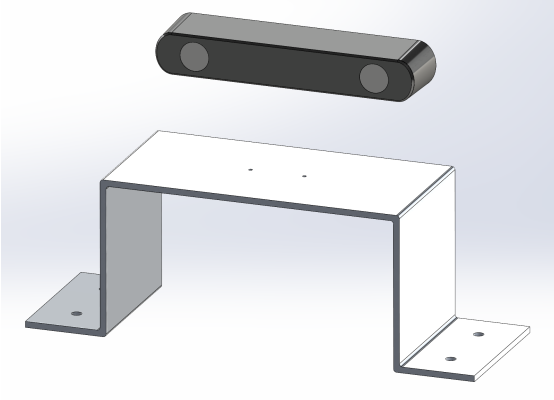


Figure 8. Step 3: attach stereo cameras to their plates

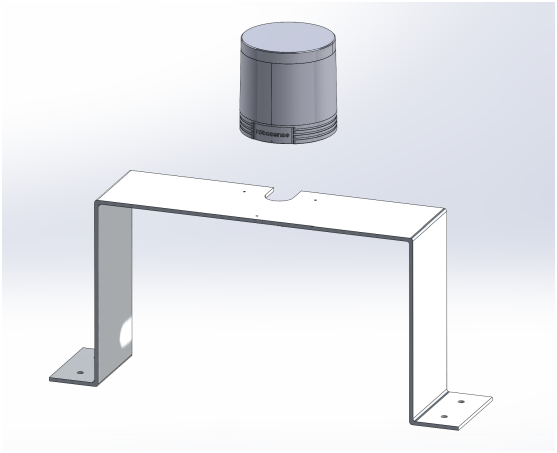


Figure 9. Step 3: attach LiDAR to its plate

Finally, screw every plates to the big one .

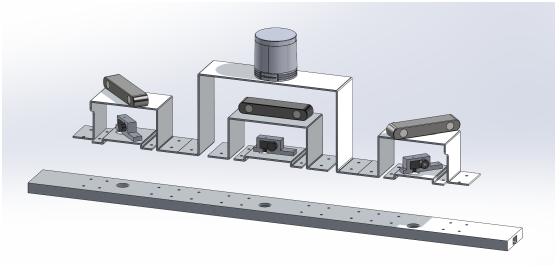


Figure 10. Step 4: attach all plates to the main one

You can now attach your main plate to the roof bar by removing the existing one but keeping the roof rack attachment on top of the doors of the car. The main plate has a hole as shown in Figure **??**with the right size to be installed with existing roof rack attachments.

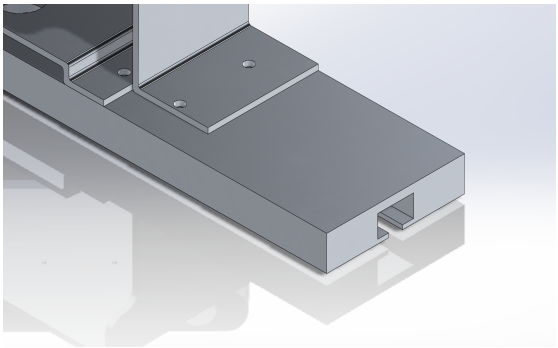


Figure 11. Step 5: attach main plate to roof rack attachment

5 MULTI-SENSORS CALIBRATION

5.1 How to install the package

To use the package, you will need to install few python packages:

- pynput and typing to use keyboard,
- cv_bridge and numpy to manage images,
- open3d and sensor_msgs_py to manage point clouds,

To have a multi sensors calibration done, the package has 5 programs:

- save_image.py and save_pointcloud.py: save images and point clouds,
- open_modify_image.py and open_modify_pointcloud.py: find the 2D key points in the images and the 3D key points in the point cloud,
- calibration.py: get the extrinsic parameters associated with this key points using RANSAC algorithm. Then visualize the results of your calibration,
- calib.py: a library to store all the utility functions the other programs use.

The package is really modular, you can modify one of the script used at a step without corrupting all your program. Moreover, if you want to make modifications, you can work on different parts of the package with your colleagues at the same time.

5.2 Steps to calibrate your system

What you should have done before starting this part:

- set up your mechanical system with all your sensors. You can find an example in the part 3 "SolidWorks model" of this report. The sensors should not move during all the calibration and in your future experiments. If they have moved, you should restart the entire calibration,
- install the ROS2 packages for your camera and your LiDAR. You can find the example of the components we have used in part 1: "Installation" and how to interface to them with ROS2 in part 2 "How to use the components",
- install properly the ROS2 package named "calibration_pkg" as described in the precedent subsection.

The calibration is done in 3 steps. The steps are exactly the same for the image and the point cloud, you just have to use the correct program depending on if it's an image or a point cloud.

5.2.1 Save your image and point cloud

- Modify the path of the folders in the script where the images and the point clouds need to be saved on your computer,
- modify the names of topics in the creation of subscribers,

```
self.image_subscriber = self.create_subscription(Image, '/zed2i/zed_node/right/image_rect_color', self.callback, 10)
self.path = "/home/premove/ros2_ws/save_image/image_" + str(self.num) + ".jpg"
```

Figure 12. Example of parameters to modify for image

```
self.lidar_subscriber = self.create_subscription(PointCloud2, '/rslidar_points', self.callback, 10)
self.path = "/home/premove/ros2_ws/save_pointcloud/pointcloud_" + str(self.num) + ".pcd"
```

Figure 13. Example of parameters to modify for point cloud

- build the package with your modification. Then, launch save_image.py and save_pointcloud.py,

```
cd ros2_ws
source /opt/ros/<your_ros2_distro>/setup.bash
colcon build --packages-select calibration_pkg
source install/setup.bash
ros2 launch calibration_pkg save_image.launch.py
    or
ros2 launch calibration_pkg save_pointcloud.launch.py
```

- You can now open a visualization window by clicking on "V" key,
- Put an object with the field of view of your camera and you LiDAR, for example a cardboard with a uniform color which contrast with the one of your background. This way it will be easier to differentiate corners of the cardboard from the background,
- During this step, take care of making room around your object. The more space around your object, the easier it will be for you to manage the point cloud in the next step,
- When you are satisfied with the pose of your object. Click on the "S" key of your keyboard to save the image or point cloud. You can find it in the folder you mentioned in your path. You have to try different poses, close and far from your sensors, with your object translated and rotated in different parts of your field of view. To have a good calibration, it is recommended to save at least 3 different images/point cloud. The more you have, the better your calibration will be,
- Tip1: if you don't have a tripod to stabilize your cardboard, open both save_image.py and save_pointcloud.py at the same time. When you will press "S" both image and point cloud will be saved at the same time,
- Tip2: each time you save an image, the name of the image saved will change according to their number. Example "image_1.png" ->"image_2.png". And until you stop your program, the last image will be upgraded so be sure to stop the program before moving your cardboard.

5.2.2 Find your 2D key points of the images and the 3D key points of your point cloud

- Modify the path of the folders in the script where the images and the point clouds where saved in the former step. You can take Figure 18 and Figure 15 as an example.

```
# Image
self.path = "/home/premove/ros2_ws/save_image/image_5.jpg"
```

Figure 14. Example of parameters to modify into open_modify_image.py

```
# Pointcloud
self.path = "/home/premove/ros2_ws/save_pointcloud/pointcloud_1.pcd"
```

Figure 15. Example of parameters to modify into open_modify_pointcloud.py

- build the package with your modification. Then, launch open_modify_image.py and open_modify_pointcloud.py

```
cd ros2_ws
source /opt/ros/<your-distro>/setup.bash
colcon build --packages-select calibration_pkg
source install/setup.bash
ros2 launch calibration_pkg open\_modify\_image.launch.py
    or
ros2 launch calibration_pkg open\_modify\_pointcloud.launch.py
```

- You can now open a visualization window by clicking on "V" key. You can see a window displaying the point cloud or image you saved,

- Click on ”M” key to modify the points studied. Your goal is to minimize the zone of research and then click on ”P” key to print the points. You can see the difference in Figure 16 and Figure 17. If you reduce your searching zone too quickly and have no points left to study the script may crash. So, be cautious when you are working with small zones.

Figure 16. Example of image with all the key points

Figure 17. Example of image with a reduced zone of key points

- If it is hard for you to focus on few points with the current resolution, open another shell and write:

```
cd ros2_ws
source /opt/ros/<your-distro>/setup.bash
ros2 param set /open_modify_image_node shift 1
    or
ros2 param set /open_modify_image_node shift 0.1
```

- The value of your image parameter needs to be a positive integer and the value of your point cloud parameter needs to be a float, it can be negative or positive,
- Note down the coordinates you got and be care to study the same point in your image and your point cloud. They will be linked between themselves in the next step.

5.2.3 Get extrinsic parameters and visualize your resulting calibration

- Modify the intrinsic parameters as in Figure 18 of your camera in your script. If you use the ZED2i camera, the zed_wrapper package we detailed in part 2 "How to use components" allow you to find them by launching the program and echoing camera_info,

```
First shell:
cd ros2_ws
source /opt/ros/<your-distro>/setup.bash
ros2 launch zed_wrapper zed2i.launch.py

Second shell:
cd ros2_ws
source /opt/ros/<your-distro>/setup.bash
ros2 topic echo /zed2i/zed_node/rgb/camera_info
```

```
# GET EXTERNAL PARAMETERS : ROTATION AND TRANSLATION VECTORS (rvec & tvec)
self.intrinsic_param = 443.2128, 258.8821, 364.5105, 370.2965 # cx, cy, fx, fy
self.dist_coeffs = np.array([0,0,0,0,0], dtype=np.float32) # the image is already rectified

objectPoints = np.array([[1.21,1.43,0.18],[1.58,0.49,-0.22],[1.72,0.37,-0.15],[2.0,0.0,0.24],[1.34,-0.71,-0.21],[1.29,-0.87,-0.17]],dtype="float32")
imagePoints = np.array([[50,167],[364,290],[400,264],[466,192],[681,304],[726,289]],dtype="float32")
```

Figure 18. Example of parameters to modify into calibration.py

- As we subscribe to the rectified image of ZED2i camera, the distortion coefficients of your camera are 0,0,0,0,0. If you use another components or you subscribe to another topic, change them,
- Add the coordinates of your 3D points in the list objectPoints and the coordinates of your 2D points in the list imagePoints. The coordinates k of your list objectPoints must represent the same point as the coordinates k of your list imagePoints. If you took points from different images, you can add them to the list without any problem as long as your components are in the same positions,
- When you are ready, launch your node to visualize the results of your program,

```
cd ros2_ws
source /opt/ros/<your-distro>/setup.bash
colcon build --packages-select calibration_pkg
source install/setup.bash
ros2 launch calibration_pkg calibration.launch.py
```

- You have now finished your calibration. If the results are not perfect, add other points in different positions and place of your field of view. Don't forget to rotate your object, put it in the corners of your Field of view, at the top or at the bottom of your image. A result of a bad calibration and a better one can be seen in Figure 19 and Figure 20.

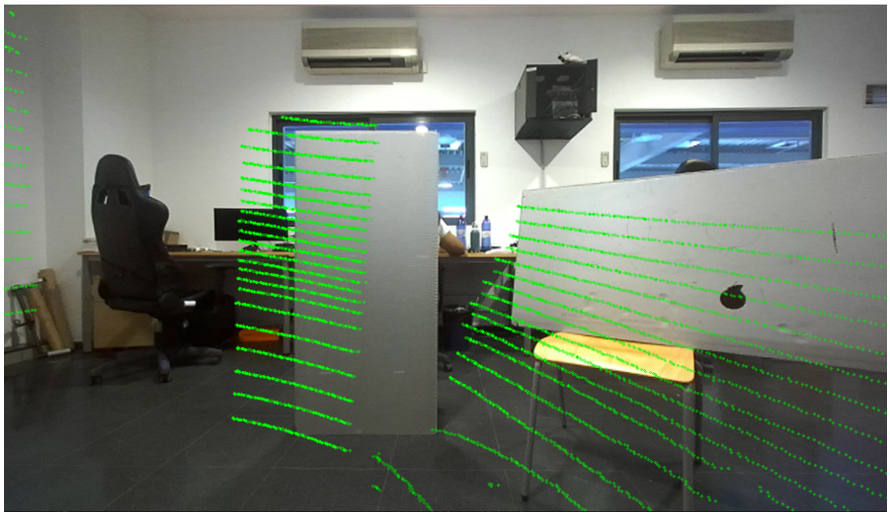


Figure 19. Example of a bad calibration

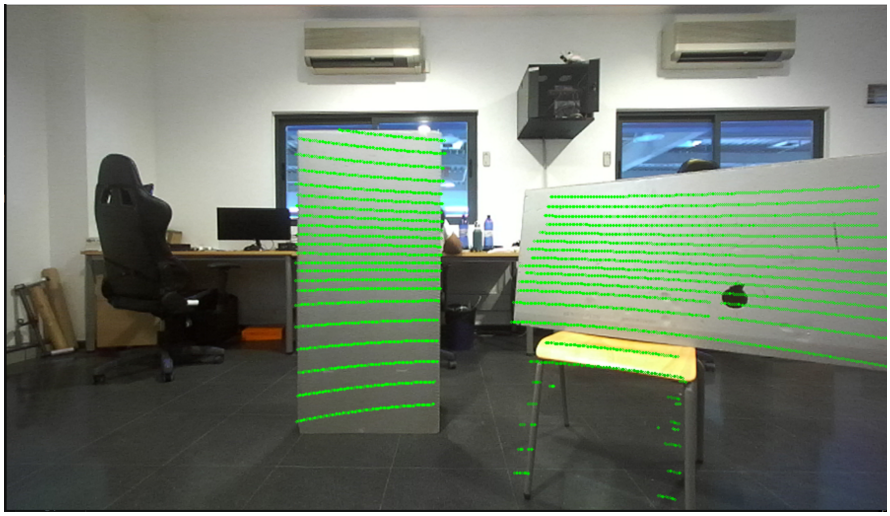


Figure 20. Example of a better calibration

5.3 How to calibrate thermal cameras

Stereo cameras create ROS2 image messages, so you can basically use the exact same process as for RGB cameras. As finding key points is done semi-manually, you can find the key points and you won't have algorithm problems.

To make it easier to find key points, we advice you to put your object in a space without objects to easily find LiDAR points and with a uniform background of a different color than the object to easily find the RGB ones. In the case of thermal cameras, these two aspects do not matter as only temperature is detected. So the best option is to find a plate which can be warmed. The easiest way to do it is by putting a metal plate in hot water but you will have to be quick to save your images as the temperature will decrease. The best way is to have a self regulated plate but its more expensive and tricky to create it.

In a cool room, humans are easily recognizable from the background thamks to their heat. The average external body temperature of a human is between 20 and 25 degrees so its not mandatory to heat the plate more than that and risk to burn yourself while moving it.

6 SOURCES AND HOW TO GO FURTHER

- This youtube video is a good introduction to camera calibration concepts: [3].
- This other video shows a working calibration: [6].
- A very precise work written on GitHub that can give ideas on how to go further in camera calibration like adding sliders for feature extraction and using automatic normal finding to get external parameters. [7]
- The papers this work is based on: [11] and [5].

To improve the results, the user needs to attach the final system in order to be sure its measures won't be altered. To be in the best conditions, it needs to use a tripod to fix the cardboard in the air. It will be easier to create different positions without moving.

For an algorithmic point of vue, it would be interesting to add sliders during the key points finding part. Indeed, doing it with the keyboard can take few minutes. If the user wants lots of measures, it can take too much time.

To find extrinsic parameters quicker, it would be better to do it automatically by letting the program finding them by itself. To do it, you can create a program which finds the normal passing by the center of the plane in the 3D point cloud and in the 2D image. After it, the program should find the rotation and translation vectors by minimising the position of these two vectors. Implementing it could take between one and three weeks to do depending on the level of the developer on this subject but it would be the better implementation to complete this package.

Obviously, to calibrate multiple sensors is not an end by itself, this package can be a part of a larger project of autonomous driving for autonomous cars or AGV. Using Yolo or TensorFlow can be the next part to merge information and analyse the environment to find the and the color of other vehicle and pedestrians with thermal and RGB camera. To find redlights or road lines with RGB camera and volumes with LiDAR.

REFERENCES

[1] askubuntu. URL: https://askubuntu.com/questions/1145811/error-installing-libjasper-dev-and-libpng12-package-in-ubuntu-19-04?rq=1. (accessed: august 2023).

[2] ROS components. *Buy Helios 5515*. URL: https://www.roscomponents.com/en/lidar-laser-scanner/312-rs-helios-5515-.html. (accessed: august 2023).

[3] First Principles of Computer Vision Channel. *Intrinsic and Extrinsic Matrices — Camera Calibration*. URL: https://www.youtube.com/watch?v=2XM2Rb2pfyQ&t=3s. (accessed: august 2023).

[4] CUDA. *CUDA install*. URL: https://developer.nvidia.com/cuda-downloads?target_os=Linux&target_arch=x86_64&Distribution=Ubuntu&target_version=20.04&target_type=deb_local. (accessed: august 2023).

[5] Carlos Guindel Jorge Beltrán. *Automatic Extrinsic Calibration Method for LiDAR and Camera Sensor Setups*. URL: https://arxiv.org/pdf/2101.04431.pdf. (accessed: august 2023).

[6] Michael May. *Lidar Camera Calibration*. URL: https://www.youtube.com/watch?v=x5HLYL_B65g. (accessed: august 2023).

[7] maym86. *ros lidar camera*. URL: https://gitlab.acfr.usyd.edu.au/sverma/cam_lidar_calibration. (accessed: august 2023).

[8] *PREMOVE project*. URL: https://www.uma.es/robotics-and-mechatronics/info/138109/premove/. (accessed: august 2023).

[9] LTD) RoboSense (Suteng Innovation Technology Co. *RS-Helios*. URL: https://www.robosense.ai/en/rslidar/RS-Helios. (accessed: 07.12.2023).

[10] ROS. *Humble debian installation*. URL: https://docs.ros.org/en/humble/Installation/Ubuntu-Install-Debians.html. (accessed: august 2023).

[11] Ariche Salma. *Object Detection and Distance Estimation via Lidar and Camera Fusion for Autonomous Driving*. URL: https://www.researchgate.net/publication/359607594_Object_Detection_and_Distance_Estimation_via_Lidar_and_Camera_Fusion_for_Autonomous_Driving. (accessed: august 2023).

[12] sfalexrog. *seekthermal$_c$amerapackage*. URL: https://github.com/sfalexrog/seekthermal_camera. (accessed: august 2023).

[13] StereoLabs. *Zed2i*. URL: https://www.stereolabs.com/zed-2i/. (accessed: july 2023).

[14] Stereolabs. *zed$_s$dk*. URL: https://www.stereolabs.com/developers/release/. (accessed: august 2023).

[15] Stereolabs. *zed$_w$rapperROS2package*. URL: https://github.com/stereolabs/zed-ros2-wrapper. (accessed: august 2023).

[16] syneart. URL: https://gist.githubusercontent.com/syneart/3e6bb68de8b6390d2eb18braw/OpenCV3.2withContrib.sh?fbclid=IwAR2uYKMEiVSCMom-KfYFPUw2ZbwvKziqvv-Y6mto9rkrnG6Btq1Cjrf_Plc. (accessed: august 2023).

[17] Seek thermal. *Compact Pro*. URL: https://www.thermal.com/compact-series.html. (accessed: july 2023).

[18] BELIER Titouan. *Multi-sensors calibration ROS2 package*. URL: https://github.com/Titou2222/calibration_pkg. (accessed: august 2023).

[19] BELIER Titouan. *Seek thermal ROS2 package*. URL: https://github.com/Titou2222/seek_thermal_ROS2. (accessed: august 2023).

RAPPORT D'EVALUATION
ASSESSMENT REPORT

Merci de retourner ce rapport par courrier ou par voie électronique en fin du stage à :
At the end of the internship, please return this report via mail or email to:

ENSTA Bretagne – Bureau des stages - 2 rue François Verny - 29806 BREST cedex 9 – FRANCE
☎ 00.33 (0) 2.98.34.87.70 / stages@ensta-bretagne.fr

I - ORGANISME / *HOST ORGANISATION*

NOM / *Name* Universidad de Málaga. Escuela de Ingenierías Industriales

Adresse / *Address* C/ Doctor Ortiz Ramos S/N. CP 29071. Málaga

Tél / *Phone (including country and area code)* +34 951 952 323

Nom du superviseur / *Name of internship supervisor*
Jesús Morales Rodríguez

Fonction / *Function* Profesor Titular de Universidad

Adresse e-mail / *E-mail address* jesus.morales@uma.es

Nom du stagiaire accueilli / *Name of intern* Titouan Belier

II - EVALUATION / *ASSESSMENT*

Veuillez attribuer une note, en encerclant la lettre appropriée, pour chacune des caractéristiques suivantes. Cette note devra se situer entre **A (très bien)** et **F (très faible)**
*Please attribute a mark from **A (excellent)** to **F (very weak).***

MISSION / *TASK*

❖ La mission de départ a-t-elle été remplie ? (A) B C D E F
Was the initial contract carried out to your satisfaction?

❖ Manquait-il au stagiaire des connaissances ? ☐ oui/*yes* ☒ non/*no*
Was the intern lacking skills?

Si oui, lesquelles ? / *If so, which skills?* _____

ESPRIT D'EQUIPE / *TEAM SPIRIT*

❖ Le stagiaire s'est-il bien intégré dans l'organisme d'accueil (disponible, sérieux, s'est adapté au travail en groupe) / *Did the intern easily integrate the host organisation? (flexible, conscientious, adapted to team work)*
(A) B C D E F

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here*_____

COMPORTEMENT AU TRAVAIL / *BEHAVIOUR TOWARDS WORK*

Le comportement du stagiaire était-il conforme à vos attentes (Ponctuel, ordonné, respectueux, soucieux de participer et d'acquérir de nouvelles connaissances) ?

Did the intern live up to expectations? (*Punctual, methodical, responsive to management instructions, attentive to quality, concerned with acquiring new skills)?*

A B C D E F

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* _____

_____

INITIATIVE – AUTONOMIE / *INITIATIVE – AUTONOMY*

Le stagiaire s'est –il rapidement adapté à de nouvelles situations ? A B C D E F
(Proposition de solutions aux problèmes rencontrés, autonomie dans le travail, etc.)

Did the intern adapt well to new situations? A B C D E F
(eg. suggested solutions to problems encountered, demonstrated autonomy in his/her job, etc.)

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* _____

_____

CULTUREL – COMMUNICATION / *CULTURAL – COMMUNICATION*

Le stagiaire était-il ouvert, d'une manière générale, à la communication ? A B C D E F
Was the intern open to listening and expressing himself /herself?

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* _____

_____

OPINION GLOBALE / *OVERALL ASSESSMENT*

❖ La valeur technique du stagiaire était : A B C D E F
Please evaluate the technical skills of the intern:

III - PARTENARIAT FUTUR / *FUTURE PARTNERSHIP*

❖ Etes-vous prêt à accueillir un autre stagiaire l'an prochain ?

Would you be willing to host another intern next year? ☒ oui/*yes* ☐ non/*no*

Fait à _____, le _____
In málaga, on 20/09/2023

Signature Entreprise _____ Signature stagiaire
Company stamp _____ *Intern's signature*

Merci pour votre coopération
We thank you very much for your cooperation