PREPARATION ET PRESENTATION D'UN ROBOT BATHYMETRIQUE POUR L'EXPOSITION « LA MER XXL »

Quentin Cardinal

ENSTA Bretagne

Superviseur : Simon Rohou

Tuteur : Simon Rohou

Adresse : ENSTA Bretagne, 2 Rue François Verny, 29200 Brest

Résumé :

Ce stage avait pour but de poursuivre le projet DeRoBat commencé en 4.4, la finalité du projet étant de faire valoir les compétences robotique et hydrographique de l'école du 29 juin au 10 juillet pour l'exposition “La Mer XXL” à Nantes.

L'objectif du projet était donc de concevoir un système autonome pouvant faire la bathymétrie d'un bassin tout en affichant en temps réel les mesures effectuées.

Le système se décompose donc en plusieurs sous-systèmes qui étaient déjà opérationnel ou proche de l'être lors du début de mon stage. Il y'avait, la récupération des données du sonar et l'affichage de celles-ci, l'asservissement du robot, le positionnement par caméras, le contrôle du moteur et de la gouverne et la communication entre les différentes parties.

Concernant les données du sonar le programme servant à la récupération de celles-ci était déjà fonctionnel, il me restait alors à gérer un affichage clair pour les visiteurs de l'exposition. Il y'avait déjà un asservissement de réaliser seulement la taille de la piscine de l'ENSTA ne permettait pas de faire de tester l'asservissement de manière correct, il a donc fallu attendre d'être à Nantes pour pouvoir la tester correctement et y effectuer quelques changements. Afin de connaître la position du bateau en temps réel nous avons fait le choix de mettre deux codes ArUco sur le bateau, grâce à la bibliothèque openCV nous pouvions avoir la position et le cap du bateau en temps réel, nous utilisions des caméras de type fish-eye, il m'a suffi de corriger la déformation créer par celles-ci afin d'avoir une position correcte.

Après avoir réglé tous les problèmes ayant pu survenir le bateau correspondait à la demande faite par le client.

Table des matières

Introduction :

Les océans sont une zone encore largement inconnue, difficile d'accès et demandent des moyens importants, aussi bien humains que technologiques, pour les étudier. Cependant, leur connaissance est un atout non négligeable pour l'économie, l'écologie mais aussi l'archéologie. Afin d'explorer facilement ces endroits, des sonars sont souvent utilises car le son est un outil efficace pour déterminer des distances entre la source et un objet. Cette méthode d'exploration s'appelle la bathymétrie et permet d'obtenir une carte 3D des environs. C'est à ces fins que sont utilisées les technologies adaptées au domaine de la bathymétrie, qui regroupe toutes les compétences de mesure de la profondeur d'une étendue d'eau pour en déterminer la topographie du fond. On pense notamment aux recherches de La Cordelière effectuées cet été par le DRASSM et l'ENSTA Bretagne qui utilisaient à la fois une automatisation de certains bateaux et des capteurs spécialisés dans la bathymétrie et la recherche sous-marine.

Les drones bathymétriques de surface présentent ainsi des perspectives intéressantes pour le futur des campagnes bathymétriques. En effet, la généralisation de l'utilisation des drones dans des domaines extrêmement variés commence depuis quelques années à intéresser les acteurs de ce secteur dans la mesure où leurs systèmes actuels sont contraignants en termes de personnels et moyens nécessaires à mettre en œuvre et/ou ne permettent pas d'accéder aux zones où la hauteur d'eau est faible. De plus, de tels engins sont facilement exploitables par des particuliers ce qui leur permet de vérifier à moindre coût l'état de leurs systèmes de retenues d'eau, dans le cas de grandes exploitations agricoles par exemple.

C'est dans ce contexte qu'a lieu l'exposition "La Mer XXL" en Juillet 2019. Cette exposition a pour but de réunir particuliers et industriels au sein d'une présentation générale des moyens actuellement mis en œuvre pour l'étude des océans. Notre projet s'inscrit dans une collaboration SHOM-ENSTA Bretagne pour présenter diverses technologies. Notre but est de concevoir et construire un bateau capable d'évoluer en autonomie dans un bassin et d'effectuer une bathymétrie de celui-ci.

Figure 1 Stand le Son pour Voir la Mer

Cependant, l'autonomie demande un positionnement du bateau au sein de son environnement, or l'exposition se déroule dans un hall, ce qui interdit l'utilisation d'un GPS. Nous avons donc exploré diverses solutions en essayant de satisfaire le cahier des charges qui nous a été proposé.

Le projet :

Présentation des objectifs :

Le but de ce projet est donc de montrer comment on se sert du son pour faire des cartes marines. Pour cela nous avons à notre disposition un bateau, le Fischkutter.

Figure 2 Démonstrateur Robotique

Ce bateau nous permet d'avoir une base déjà fonctionnelle, nous avons juste rajouter une carte Arduino et son shield Moteur afin de contrôler les moteurs et une Raspberry Pi qui sert de « cerveaux » du bateau. Elle envoie les commandes moteur et safran à l'Arduino, récupère les données du sonar pour les envoyer sur le PC distant.

Il ne nous reste plus qu'à avoir un PC qui se connecte à la Raspberry Pi via le wifi créer par cette dernière afin d'échanger les données du sonar et les commandes PWM.

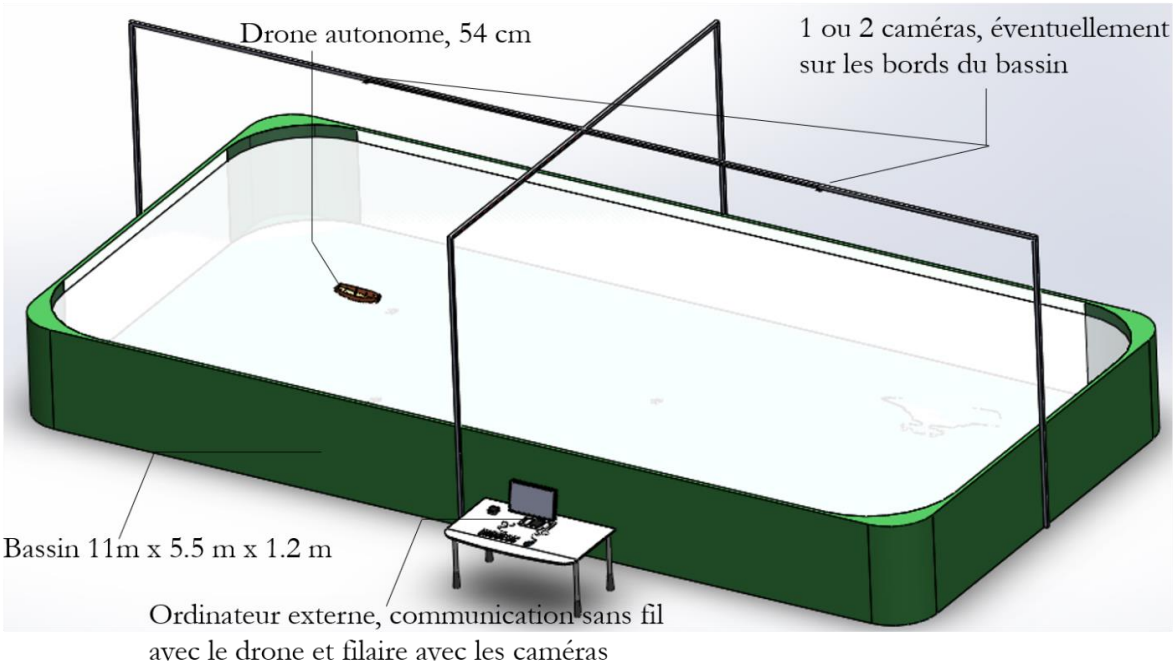

Figure 3 Bassin "La Mer XXL"

Planification :

Ce projet fait suite au travail effectué en UV 3.4 et 4.4, à la fin de ces deux UV la partie hardware et la partie communication étaient finit et complète. Il ne me restait plus qu'a faire la régulation du bateau et finaliser le positionnement et l'affichage de la bathymétrie.

- Du 5 au 26 juin :

Dès le début du stage je me suis attelé à avoir le meilleur positionnement possible. En effet, Si la position du bateau n'est pas connue alors la régulation et donc la démonstration deviennent impossible.

Il m'a ensuite fallu m'occuper de l'affichage de la bathymétrie afin d'avoir quelque chose de représentatif tout en restant compréhensible des visiteurs.

Après être sûr d'avoir une bathymétrie et un positionnement correct j'ai pu m'attaquer à la régulation, cette partie m'a pris plus de temps que je le pensais, en effet il m'était très compliqué de tester mon programme dans la piscine de l'ENSTA Bretagne. Etant donnée qu'elle ne fait que 3*4 m le bateau n'avait pas le temps de rejoindre la ligne à suivre avant d'arriver au bout de cette dernière.

J'ai passé les derniers jours avant l'exposition à préparer et regrouper tout le matériel dont on pourrait avoir besoin et prendre contact avec les gérants du stand afin qu'on s'organise pour installer le matériel nécessaire.

- Du 26 juin au 10 juillet :

Avec Philibert Adam nous somme arriver sur le stand le mercredi 26 juin 2019. Nous en avons profité pour nous installer et préparer la démonstration du samedi. Du samedi 29 juin au mercredi 10 juillet nous avons fait plusieurs démonstrations sur le bassin à l'aide du bateau et des Rov de l'ENSTA. Notre rôle était de présenter la démonstration et de parler plus en détails de celle-ci ou de l'école aux visiteurs le souhaitant.

- Du 10 juillet au 30 aout :

Après avoir passé 2 semaines à « La Mer XXL » nous avons passé la semaine à débriefé ce qu'on avait pensé de l'exposition, de ce qui serait à améliorer et ce qui était bien. Après la fermeture de l'ENSTA (du 19 juillet au 12 aout) j'ai repris le travail qu'y avait été fait à Nantes afin de pouvoir le faire lors de la présentation de l'ENSTA et de ses spécialités le vendredi 30 aout. J'en ai profité pour faire une documentation claire et précise sur comment fonctionne le bateau et les différents programmes et j'ai fait en sorte de pérenniser la démonstration dans le temps (code ArUco en PVC…)

Le système :

Comme précisé plus haut, je n'ai modifié que la partie localisation, régulation et l'affichage de la bathymétrie, j'ai malgré cela, du m'apprivoiser avec les deux autres parties qu'étaient la communication entre le bateau et le pc centralisant les données et le hardware déjà faite.

Communication et Hardware :

➢ *Hardware :*

Le bateau choisit pour faire ce projet est le Fischkutter de Carson, ce bateau présente l'avantage d'avoir une architecture électrique déjà présente et fonctionnelle. Le but était donc de démonter le bateau afin d'y ajouter nos composants (Raspberry Pi et Arduino, Shield Moteur).

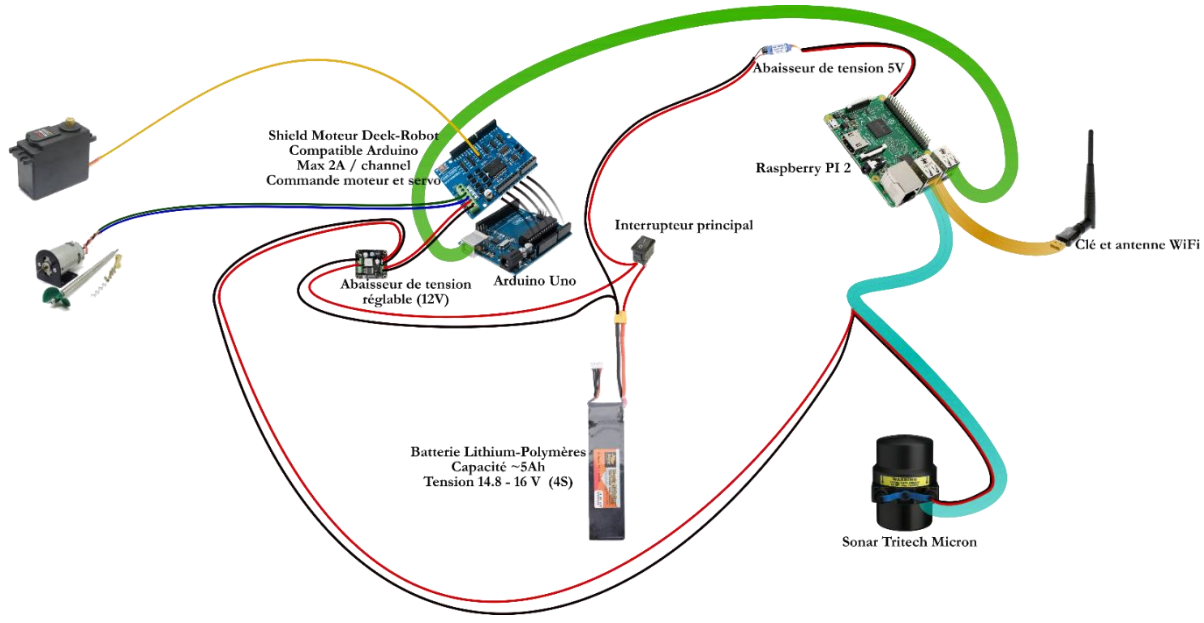

Figure 4 Hardware du bateau

➢ *Communication :*

Pour l'architecture logicielle je n'ai rien eu à changer, il m'a fallu comprendre comment elle fonctionnait afin de pouvoir y faire transiter de nouvelles données. Toutes les taches nécessitant des ressources (Calcul de la régulation, traitement d'image…) était faite depuis le PC et transmise au robot.

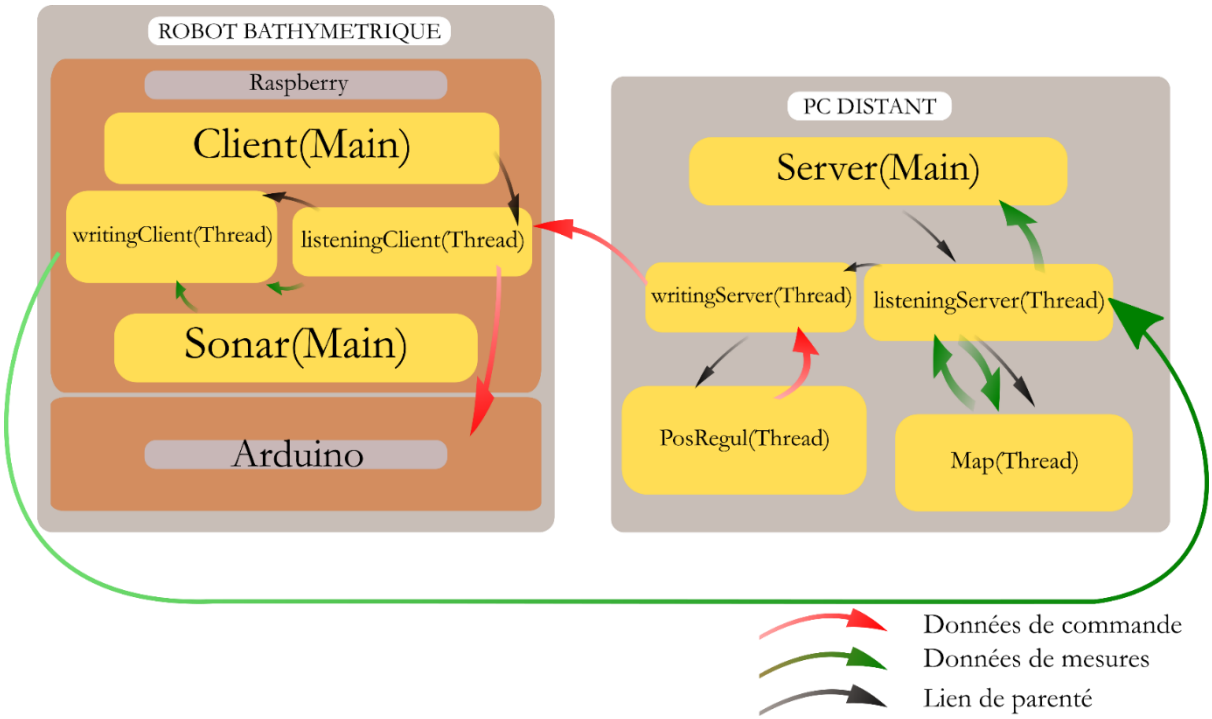

Figure 5 Architecture Logicielle

Affichage de la bathymétrie :

Nous étions dans une piscine de 10.2*5.2*1 mètre, pour simuler un fond marin nous avons fait le choix d'y poser des briques de Lego. Les briques pouvant s'empiler nous permettait d'avoir un fond modulable et facile à construire.

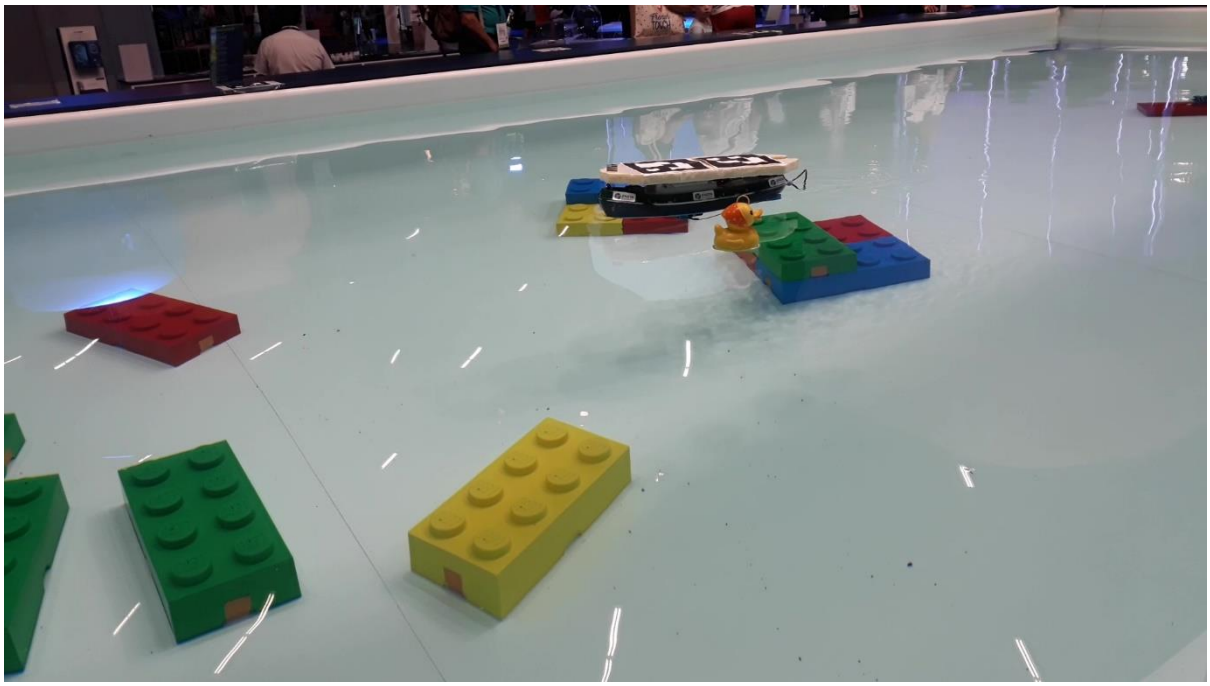

Figure 6 Fond brique de Lego

Les données du sonar était déjà traité et renvoyé sur le PC principal quand j'ai repris le projet, il ne me restait plus qu'à afficher les données.

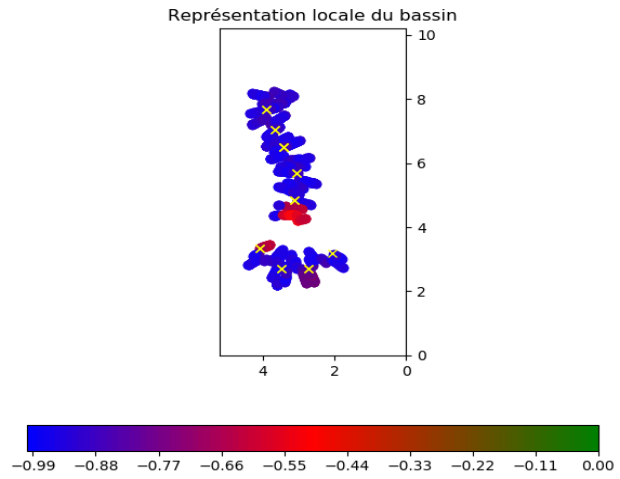

Figure 8 Représentation locale du bassin

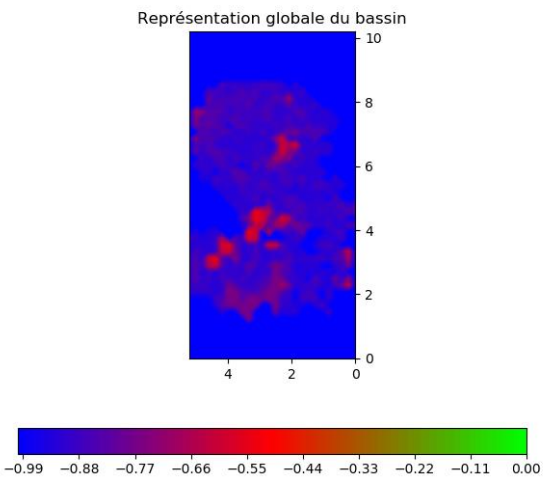

Figure 7 Représentation globale du bassin

Deux possibilités ont été envisagé concernant l'affichage, l'affichage de tous les points renvoyé par le sonar sur un scatter de Matplotlib (la bibliothèque python) ou l'affichage via la fonction contourf de Matplotlib qui nous permet d'avoir un rendu plus proche d'un Modèle Numérique de Terrain. Les deux solutions ont été retenue.

Le scatter nous permit de montrer brièvement comment on crée une cartographie marine, il se destine plus aux visiteurs n'ayant pas le temps de rester sur le stand. Pour les plus patient, le contourf leur permit de voir la carte du bassin se créer au fil du temps.

Le problème avec le scatter de Matplotlib est que l'affichage est rechargé à chaque fois que l'on veut ajouter des points de données, cette méthode d'affichage devient très lente après une à deux minutes de démonstrations. C'est pour cela que nous effaçons les données de la fenêtre Matplotlib tous les 20 paquets de données reçu.

Une autre possibilités qui fut envisagé fur d'afficher les points correspondant au données renvoyées par le sonar directement sur l'image prise par les caméras. Malheureusement je n'ai pas eu assez de temps pour mettre en œuvre cette solution.

Positionnement :

➢ *Installation :*

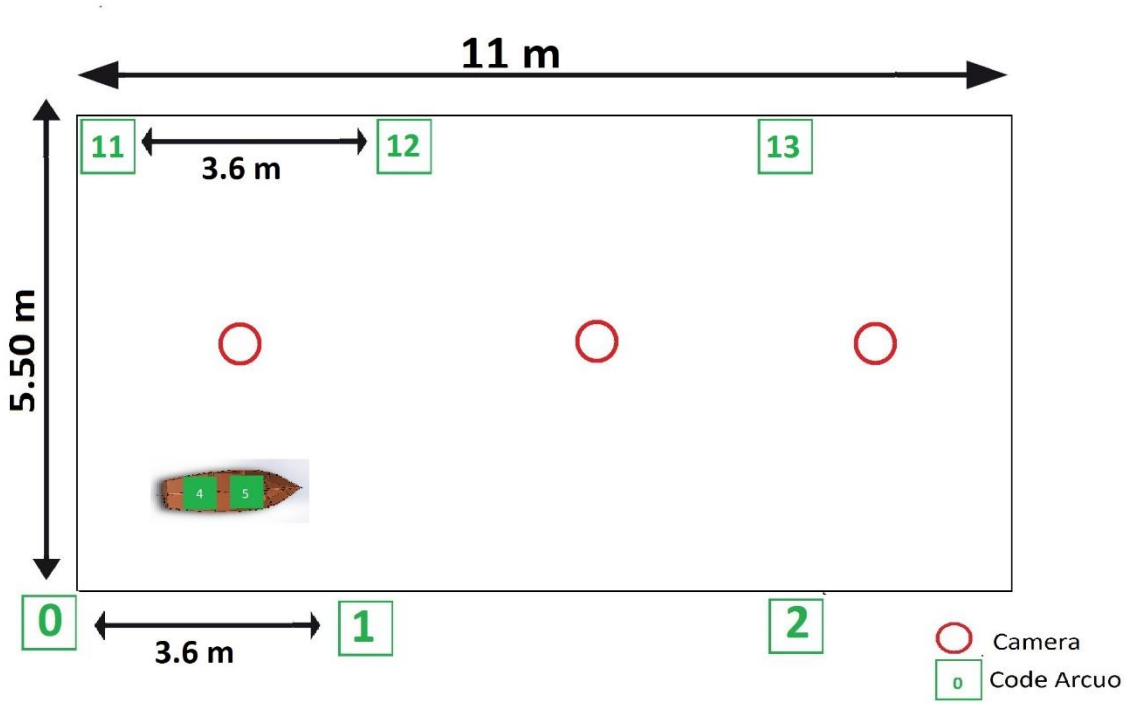

Figure 9 Plan du bassin

Etant donnée que la démonstration se déroule en intérieur l'utilisation d'un GPS est donc impossible. Pour pallier ce problème nous avons fait le choix de mettre des caméras au-dessus du bassin. Afin de couvrir efficacement tout le bassin j'ai calculé que nous aurons besoin d'en avoir trois.

Dans le but d'avoir une cohérence dans le positionnement du bateau le bassin est parsemé de code ArUco.

Il faut que les caméras se situe à environ 4 m de la surface de l'eau, nous avons donc besoin d'une structure nous permettant d'attacher les caméras à environ 5,20m du sol. Après avoir contacter le standardiste nous nous sommes mis d'accord que nous attacherons les cameras depuis les structures soutenant le toit du hall. Avec l'aide d'un manutentionnaire nous avons descendu une poutre depuis le toit afin d'y accrocher les caméras.

Figure 10 Installation des caméras

On peut voir, entouré de rouge, les 3 caméras suspendu au-dessus de la piscine.

➢ *Fonctionnement :*

Les 3 caméras sont des caméras de type fish-eye Axis M3046, elles sont alimentées et nous recevons les données via PoE. Afin de pouvoir avoir les 3 caméras en même temps nous utilisons un switch Ethernet.

Utiliser les caméras via Ethernet nous permet de faire transiter des données sur de plus grandes distance qu'en USB et nous évite la perte de données via un réseau Wifi. Pour une exposition comme « La Mer XXL » cela nous permet d'être sûr d'avoir le flux vidéo des caméras.

Avant de pouvoir détecter le bateau dans l'image des caméras il faut d'abord commencer par effectuer une « remise à plat » de l'image. En effet, comme dit plus haut les caméras sont de types fish-eye, ce qui veut dire qu'elles ont un grand angle de vue, nous aurons donc de la distorsion aux extrémités de nos images.

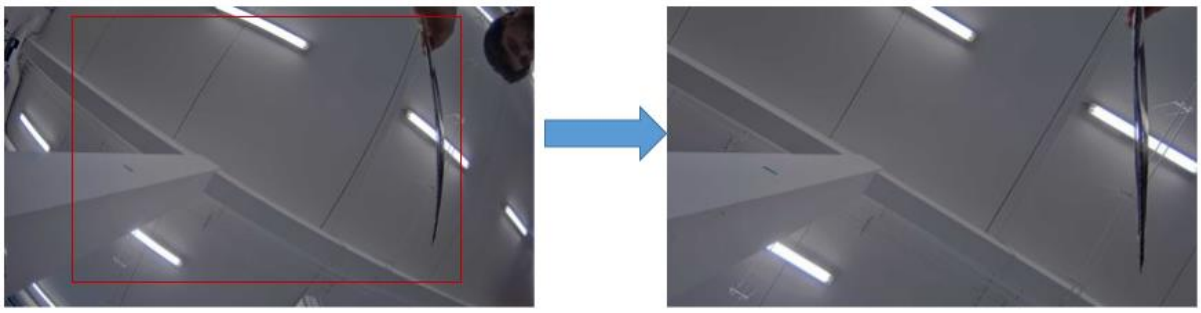

Figure 11 Calibration des caméras

On peut voir qu'a cause de cette transformation on à une perte d'information (on ne prend que le rectangle rouge) mais les lampes, le carton… sont à nouveau droit. Je dois juste m'assurer que tout le bassin dans lequel évolue le bateau soit compris dans le rectangle rouge.

Pour avoir la position du bateau dans le bassin je commence par « chercher » le bateau grâce à la fonction chooseCap.

Fonction chooseCap in newcameramtx, roi, mtx, dist, oldCap, x out: frame, corner, id, camNumber
Si oldCap == 1: Ouverture du flux vidéo de la caméra 1 cap1 Redressement de la frame fish-eye frame1 Détection des Arucos dans frame1, récupération de leurs id et corner Fermeture du flux vidéo cap1 Si 4 et 5 sont dans id : Return frame1, id, corner,1 Ouverture du flux vidéo de la caméra 2 cap2 Redressement de la frame fish-eye frame2 Détection des Arucos dans frame1, récupération de leurs id et corner Fermeture du flux vidéo cap2 Si 4 et 5 sont dans id : Return frame2, id, corner,2 Ouverture du flux vidéo de la caméra 3 cap3 Redressement de la frame fish-eye frame3 Détection des Arucos dans frame1, récupération de leurs id et corner Fermeture du flux vidéo cap3 Si 4 et 5 sont dans id : Return frame3, id, corner,3 Si oldCap == 2 : On fait la même chose qu'au-dessus Si oldCap == 3 : On fait la même chose qu'au-dessus

Cette fonction me permet de trouver les codes Aruco du robot de manière à ouvrir le moins de flux vidéo possible. Lors du lancement du programme on fera une recherche du bateau sur le bassin, ensuite après l'avoir trouvé on devinera quel flux vidéo ouvrir en fonction de sa position (camera 1,2 ou 3).

Une fois le bateau trouvé sur la bassin je renvoie ses coordonnées « en brut » au programme principale qui lui appellera la fonction getPosition et getCap qui nous renvoie respectivement la position et le cap du bateau.

Fonction getPosition in: corner, id, frame out: xBoat, yBoat

On récupère la position de l'ArUco 4 et 5 dans x4,y4 et x5,y5
x = (x4 + x5) /2
y = (y4 + y5)/2
 	
Si on a la position des ArUcos correspondant au bord du bassin : //Change selon la caméra
pos = [[x – ArUco0], // Translation de la position
[y – Aruco0]] // ArUco0 correspond à l'ArUco qui doit se situer en (0;0)
 	
pos = getRotationMatrix(getTheta)) * pos //Rotation de la position
 	
xBoat prend la valeur de pos[0,0] après une conversion de pixel vers mètre
yBoat prend la valeur de pos[1,0] après une conversion de pixel vers mètre

Dans cette partie du programme nous utilisons la fonction getRotationMatrix qui nous renvoie la matrice de rotation en fonction de thêta. La fonction getTheta est la pour calculer le thêta de cette matrice, elle prend l'angle entre les deux ArUco en début de caméra (le 0 et 10 ou le 1 et 11 ou le 2 et 12).

Fonction getCap in: corner, id, theta out: cap
On récupère la position de l'ArUco 4 et 5 dans x4,y4 et x5,y5 cap = atan2(y5 - y4, x5 - x4) – theta

➢ *Améliorations possibles :*

Concernant la détection du bateau une amélioration possible aurait été de définir la prochaine caméra en fonction de la trajectoire du robot et non de manière arbitraire comme fait ici.

On peut voir que les fonctions getPosition et getCap commence toutes deux par une récupération de la position des ArUcos 4 et 5, on aurait pu optimiser cette recherche en les prenant directement depuis la fonction chooseCap

Régulation

➢ *Principe de fonctionnement :*

Un des buts du projet était d'avoir un bateau complétement autonome, pour faire cela j'ai mis en place une régulation en cap et en vitesse du bateau.

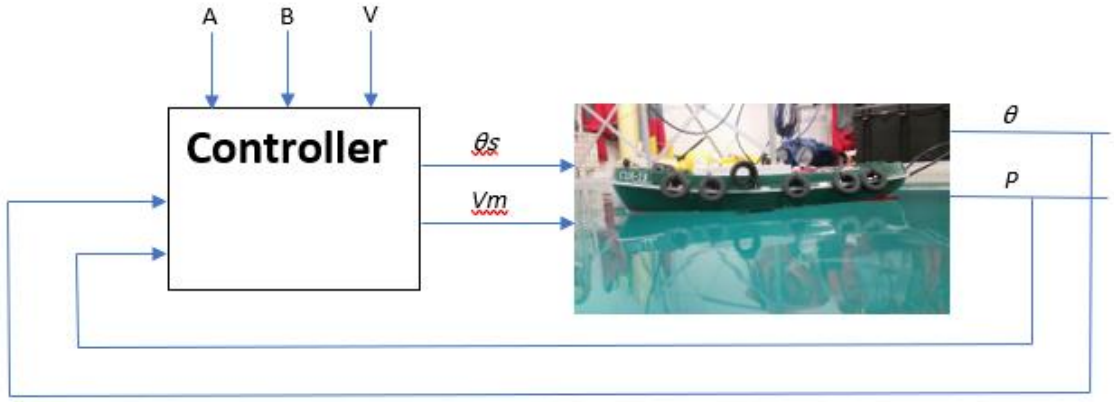

Figure 12 Contrôleur

Nous pouvons voir au-dessus comment est-ce que j'ai imaginé le contrôleur. Nous définissons A, B et V qui correspondent respectivement à la ligne à suivre (A, B) et à la vitesse désirée.

La partie la plus technique de ce contrôleur est le suivi de ligne, son but est très simple, ramener le robot sur la ligne à suivre. IL faut prendre en compte l'angle de cette ligne ainsi que l'écart à la ligne du robot afin d'avoir un programme fonctionnant correctement.

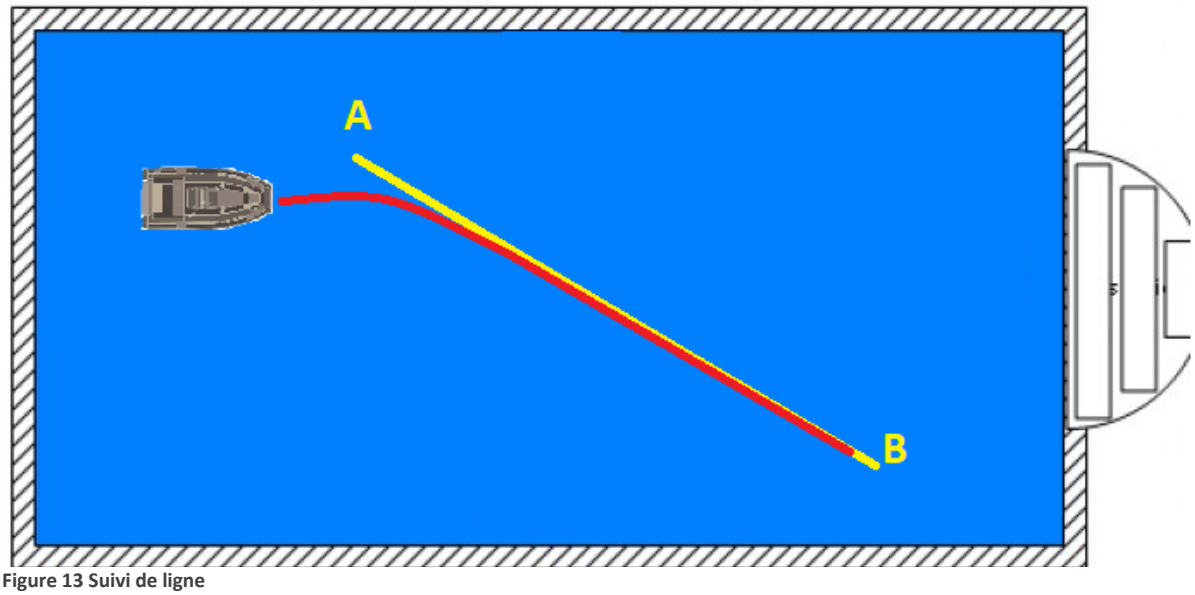

Figure 13 Suivi de ligne

➢ *Fonctionnement :*

Nous avons donc deux loi de régulation à créer pour ce bateau. La première, celle pour la vitesse est très simple, il s'agit juste d'une conversion de la vitesse désirée (en m/s) en signaux PWM pouvant être interpréter par l'ESC et donc le moteur. Pour ce faire j'ai créé ce programme :

Fonction toPWM in : commande out : self.commande
Si le bateau va en avant : gouverne = commandes[0,0] vitesse = commandes[1,0] self.commande =(175* gouverne + neutreServo, 238*vitesse + neutreMoteur)

Le neutreServo et neutreMoteur sont deux variables qui ont été définit préalablement, elles correspondent au signal PWM à envoyer pour que le servo contrôlant la gouverne et le moteur soit dans leur position dite « neutre ». C'est-à-dire la gouverne reste droite et le moteur ne tourne pas.

Concernant la gouverne du bateau, sa loi de commande est plus élaborée que celle du moteur. J'ai fait le choix d'un contrôleur proportionnel-dérivée ce qui est le contrôleur le plus commun pour ce genre de régulation. En effet pour un simple suivi de ligne le fait d'avoir ajouté la dérivée dans les calculs nous permet d'éviter d'osciller autour de notre ligne. Voici ce contrôleur :

Fonction getCommand in : X, a, b, vTarget out : commande

d = b-a
e = det(hstack((X[:2]-a, d/norm(d))))
capTarget = atan2(d[1], d[0]) + 0.5*arctan(e)
deltaTarget = capTarget - X[2]
 	
if math.cos(deltaTarget) >= 0:
commande[0,0] = -angle_max*math.sin(deltaTarget)
else:
if math.sin(deltaTarget) >= 0:
commande[0,0] = -angle_max
else:
commande[0,0] = angle_max
 	
commande[1,0] = vTarget
commande[0,0] = max(-angle_max, min(commande[0,0] , angle_max))
commande[1,0] = max(-vmax, min(commande[1,0] , vmax))

Dans ce programme on commence d'abord par définir ce qu'on appelle « l'erreur à la ligne », il s'agit de la variable e. Cette variable nous permet de savoir à quelle distance on se situe de notre ligne, le but étant de la rejoindre avec plus ou moins d'angle.

On retrouve ensuite notre cap à suivre (capTarget), ce cap est un melange entre le cap de la ligne (atan2(d[1], d[0])) et notre écart à la ligne (0.5*arctan(e)). Après avoir eu ce cap il nous reste plus qu'à soustraire le cap du bateau afin d'avoir un angle de rotation.

La spécifié de ce projet est que la gouverne du bateau ne doit pas dépasser un certain angle angle_max qui vaut π /4, c'est pour cela que l'on retrouve un « tri » qui nous empêche la commande de dépasser π /4. Nous avons le même tri concernant la vitesse.

➢ *Améliorations possibles :*

Je pense qu'avant d'essayer d'améliorer la régulation il faut commencer par l'algorithme renvoyant la position du bateau. En effet, cet algorithme reste assez lent, il faudrait trouver une solution pour augmenter la vitesse de détection du bateau dans l'image ou alors faire un filtre de Kalman entre deux images.

Conclusion

Lors de « La Mer XXL », ce robot autonome nous a permis de démontrer à un grand publique comment on peut créer une carte marine à partir d'un sonar. Pour ce genre de mission le programme du bateau est suffisamment avancé, mais il reste quelques points d'améliorations possible.

Il faudrait améliorer le positionnement du bateau, on peut, par exemple, utiliser un filtre de Kalman pour calculer la position du bateau entre deux images caméra. Les caméras feraient donc office d'amer afin de repositionner le bateau.

On peut aussi s'affranchir du système de socket avec lequel communique les différents éléments du bateau, utiliser les nœuds de Ros semble être une meilleur idée.

Du point de vue de l'organisation, ce projet m'a appris à mieux définir le temps qu'il me faut pour faire certaines tâches. De plus, le fait d'être tout seul à gérer le projet me permet d'ajuster très facilement mon emploi du temps et de travailler sur la partie la « plus urgente ».

D'un point de vue technique, grâce à ce robot autonome j'ai pu mettre en œuvre une des régulations que nous avons vu lors de l'UV 4.7 – Méthodes ensemblistes pour la robotique. De plus, le travail fait sur les images récupéré des caméras m'a permis d'utiliser les connaissances acquises avec l'UV 4.6 – Introduction au traitement numérique des images et OpenCV.

Remerciement :

Je remercie Simon Rohou de m'avoir donnée cette opportunité de stage ainsi que tout le pôle robotique pour le soutient technique qu'ils ont pu m'apporter. JE tiens aussi à remercier les hydrographes de l'ENSTA Bretagne pour l'aide sur la partie bathymétrie, et enfin je remercie Ingrid Le Toutouze pour l'organisation du stand.

Table des figures

Annexes :

Github avec le projet (pour La Mer XXL)
https://github.com/Matthix7/DeRoBat
Gihub avec le projet (Pour le bassin hydro)
https://github.com/QuentinCar/DemoHydro
Détection des ArUcos
https://docs.opencv.org/master/d5/dae/tutorial_aruco_detection.html
Régulation du bateau
https://www.ensta-bretagne.fr/jaulin/control.html
Calibration des caméras
https://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html