ÉCOLE NATIONALE D'INGÉNIEURS DE BREST

Université de Bretagne Occidentale

Rapport de stage

Master SIIA

Tracking et perceptions virtuelles pour un essaim de robot MONA

Auteur :
Tristan Guichaoua

Tuteurs :
Jérémy Rivière
Aymeric Henard

Juin 2022

Table des matières

Table des figures

Liste des tableaux

Glossaire

broadcast Envoie d'un paquet à tous les membres d'un réseau. 18

centrale à inertie Instrument utilisé en navigation, capable d'intégrer les mouvements d'un mobile (accélération et vitesse angulaire) pour estimer son orientation (angles de roulis, de tangage et de cap), sa vitesse linéaire et sa position. 5, 9

I²C Bus informatique qui permet de relier facilement un microprocesseur et différents circuits. 9

multicast Envoie d'un paquet à tous les membres d'un groupe du réseau. 18

temps-de-vol Dans le domaine des radio-fréquences : le temps mis par un signal pour aller de l'émetteur au récepteur. 13

Acronymes

CERV Centre Européen de Réalité Virtuelle. 9

ENIB École Nationale d'Ingénieurs de Brest. 6

IMU centrale à inertie (Inertial Measurement Unit). 9

IR infrarouge. 2, 8, 12

Lab-STICC Laboratoire des Sciences et Techniques de l'Information, de la Communication et de la Connaissance. 2, 6, 7

RA Réalité Augmentée. 17

SIIA Systèmes Intelligents Interactifs et Autonomes. 6

1 Introduction

1.1 Mise en contexte

Dans le cadre de ma formation à l'École Nationale d'Ingénieurs de Brest (ENIB) pour obtenir un diplôme d'ingénieur généraliste et d'un master en Systèmes Intelligents Interactifs et Autonomes (SIIA), j'ai réalisé un stage de fin d'étude auprès de l'équipe INUIT du Laboratoire des Sciences et Techniques de l'Information, de la Communication et de la Connaissance (Lab-STICC) sur 6 mois du 31 janvier au 29 juillet 2022.

Le sujet du stage est la mise en place d'un environnement pour la réalisation d'expériences sur des algorithmes comportementaux sur un essaim de robots avec l'utilisation d'un système de tracking pour suivre le mouvement des robots. J'ai réalisé, en préambule de ce stage, un état de l'art de différentes méthodes de tracking.

Cet environnement sera utilisé par un doctorant, Aymeric Henard, qui, dans le cadre de sa thèse, étudie le fonctionnement d'algorithmes comportementaux sur des essaims de robots. L'idée étant de décomposer les éléments de l'algorithme et de comprendre comment les différents paramètres peuvent influer sur le résultat. Par exemple, quels paramètres faut-il changer pour passer d'un comportement d'agrégation à un comportement répartition dans l'espace.

1.2 Lab-STICC

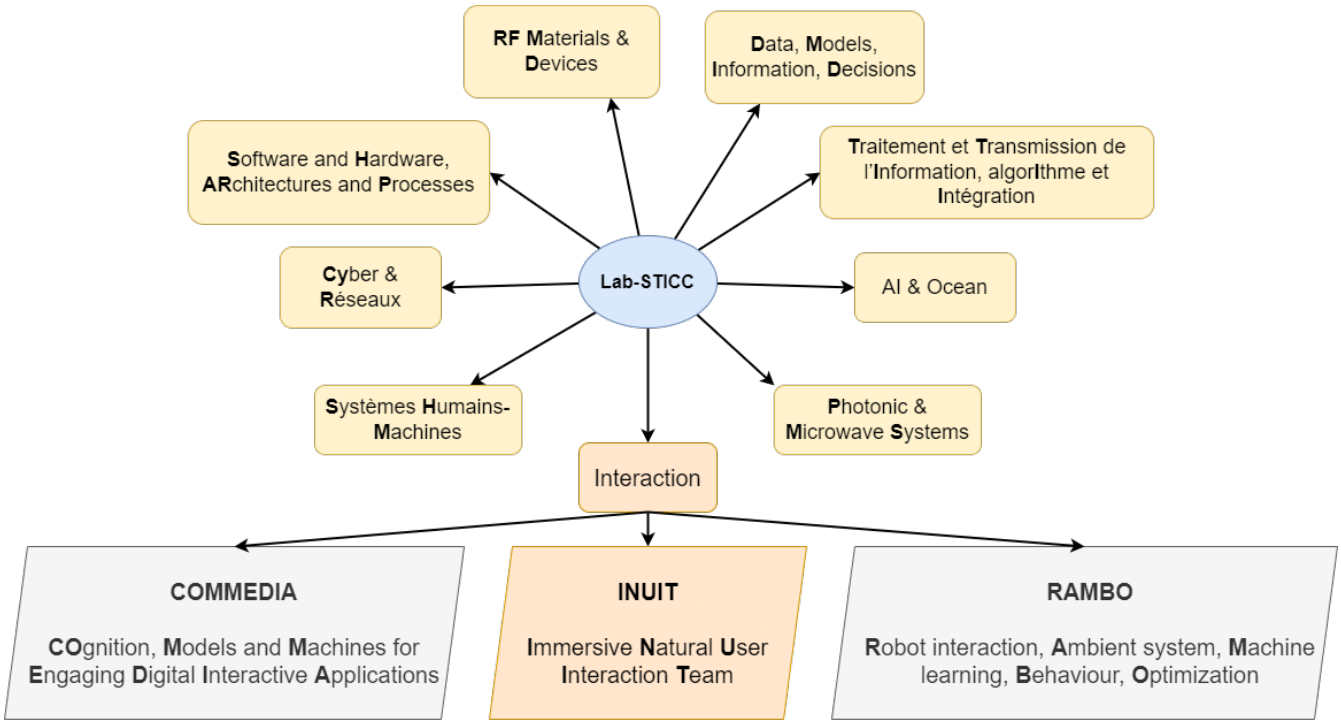


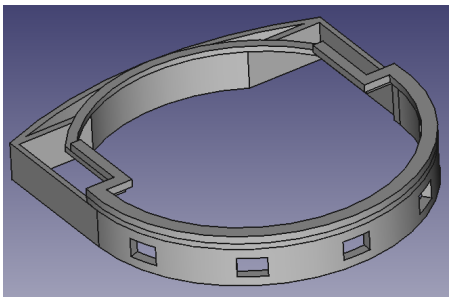
FIGURE 1.1 – Organigramme du Lab-STICC

Le Lab-STICC est une unité mixte de recherche ayant pour tutelles le Centre national de la recherche scientifique (CNRS), IMT Atlantique, l'université de Bretagne Occidentale, l'université de Bretagne Sud, l'ENSTA Bretagne et l'École nationale d'ingénieurs de Brest [4]. Il affiche une capacité avéré de couvrir un large spectre scientifique autour des sciences du numérique, et avec en particulier cette faculté d'adresser des champs disciplinaires variés (Théorie de l'Information, Ondes & Matériaux, Electronique et Informatique embarquées, Sciences des données, Communication et détection de signaux, Interfaces Homme-Machines,..) suivant des thématiques/secteurs applicatifs multiples : l'environnement maritime, les objets communicants, la défense, le spatial, la santé, la sécurité, la robotique... [5]

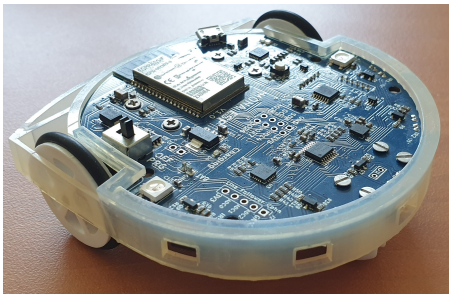
Le Lab-STICC est composé de neuf départements, eux-mêmes découpés en équipes (figure 1.1). J'ai rejoint le département interaction qui travaille sur l'interaction des systèmes et des utilisateurs. Ce département compte trois équipes : COMMEDIA qui travaille l'aspect psychologique, RAMBO qui travaille sur l'apprentissage automatique et enfin INUIT, l'équipe que j'ai rejoint, travaille sur les technologies immersives [3].

1.3 Base de travail

Ce stage fait suite à celui d'un précédant stagiaire qui a réalisé la mise en place d'une table tangible interactive avec les robots Mona (section 2.1). Le travail réalisé lors de ce stage m'a permis de prendre plus rapidement en main les robots et m'a évité des problèmes. J'ai pu réutiliser les outils mis en place pour simuler les robots Mona avec le simulateur de robots WeBots (section 2.3). Aussi, le casque qui a été créé (figure 1.2) permet d'avoir une surface entre les capteurs pour permettre à la lumière IR des capteurs d'un autre robot de s'y réfléchir. Sans ce casque, les robots ont du mal à se détecter les uns les autres.



(a) Sous FreeCAD



(b) Un robot avec le casque

FIGURE 1.2 – Casque pour les robots Mona

1.4 Objectifs

Les algorithmes de comportement en essaim nécessitent que chaque robot puisse percevoir leurs voisins immédiats. Les capacités perceptives des robots Mona qui seront utilisés pour les expériences, sont limitées. Ils ne peuvent pas percevoir leurs voisins : leurs capteurs de proximité ne peuvent pas faire la différence entre un obstacle et un robot. Mon objectif sera donc d'implémenter une solution pour augmenter la perception des robots et de percevoir leurs voisins.

Je me servirai de robots et d'un système de tracking que je décris dans le chapitre 3. Une passerelle présentée dans le chapitre 4 jouera le rôle d'intermédiaire entre les deux. Enfin, je validerai le fonctionnement de ces différents composants dans le chapitre 5.

2 Outils & Matériels

2.1 Robot Mona

Les robots Mona (figure 2.1) sont des petits robots motorisés bon marchés (~100 € / unité) utilisés principalement dans l'éducation et la recherche [2]. Ces robots sont équipés de deux moteurs pilotables individuellement et qui peuvent fonctionner en marche avant ou marche arrière, de cinq capteurs de proximité à infrarouge placés sur le devant, d'une centrale à inertie (Inertial Measurement Unit) (IMU), de deux leds RGB et d'une connectibilité Wi-Fi. Ils possèdent également un bus I^2C qui permet d'étendre les possibilités du robot en ajoutant de nouveaux composants. Leur processeur fonctionne sous Arduino, ce qui permet d'utiliser la myriade de bibliothèques Arduino, mais aussi de simplifier grandement le développement des programmes. Ces robots sont aussi *OpenSource* et *OpenHardware*, ce qui signifit que les resources techniques sont disponibles et qu'il est relativement aisé de modifier les robots.

2.2 Localisation

Le CERV est équipé d'un système de localisation : OptiTrack. Il s'agit d'un système de localisation qui utilise plusieurs caméras placées autours de la zone de localisation et qui peut suivre la position et l'orientation d' objets. De petites boules réfléchissantes servent de marqueurs. Les caméras émettent de la lumière infrarouge qui se réfléchit sur les marqueurs. Grâce aux différents points de vue des caméras, il est possible de déterminer la position tri-dimentionnelle de chaque marqueur. Pour reconnaitre un objet, on y place plusieurs marqueurs, puis on indique au logiciel que tel groupe de marqueurs correspond à tel objet. C'est ce que l'on appel un *rigid body*, un corps indéformable. La distance entre les marqueurs est toujours la même et c'est ce qui permet de les identifier.

OptiTrack permet aussi de faire de la *motion capture*. On peut enregistrer les mouvements d'une personne portant une combinaison ayant des marqueurs. La motion capture est utilisée dans les jeux vidéo ou encore dans l'animation pour obtenir des mouvement de personnages plus réalistes.

Dans le stage précédant au mien, une table interactive tangible était employée pour localiser les robots. Cependant, celle-ci est trop petite et utilise de la lumière infrarouge qui perturbe les capteurs de proximité des robots.

2.3 Simulation

WeBots est un simulateur de robots 3D gratuit et open-source utilisé dans l'industrie, l'éducation et la recherche [9].

La simulation permet de tester différentes combinaisons de robots et de capteurs avant d'avoir à les acheter. Cela permet aussi plus de fléxibilité dans le développement d'algorithme. En effet, on peut rapidement modifier le code et le déployer en un instant sur l'ensemble des robots virtuels. On à aussi un controle total sur l'environnement, ce qui permet de confronter le robot à un obstacle précis sans avoir à gérer d'éléments parasites.

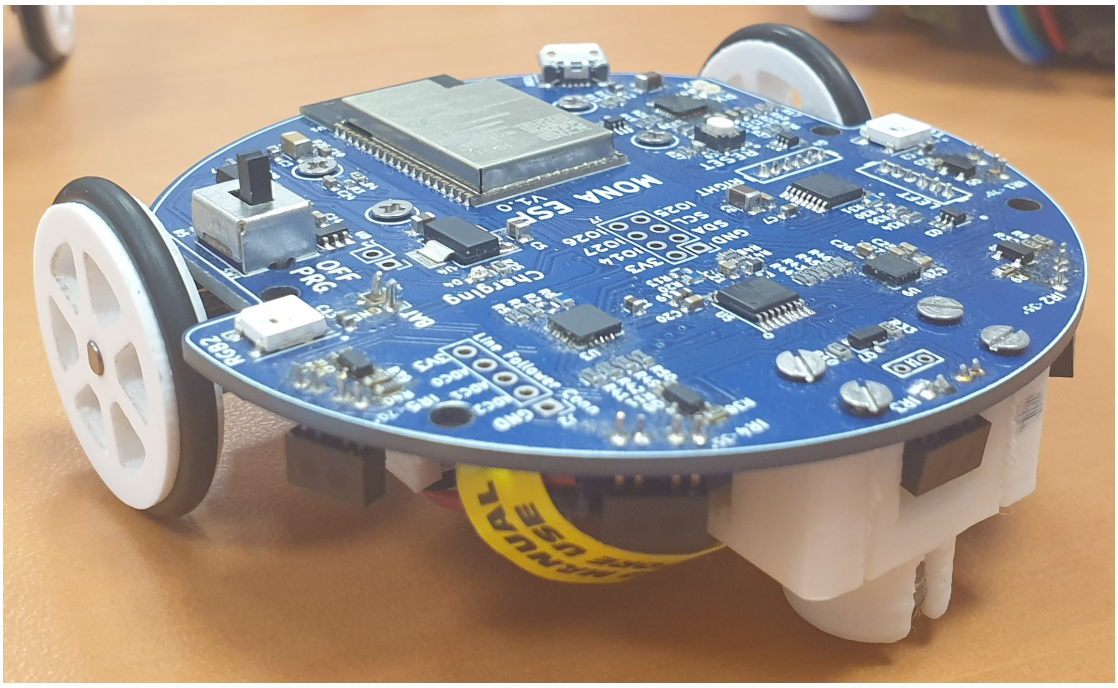


FIGURE 2.1 – Photo de robots Mona

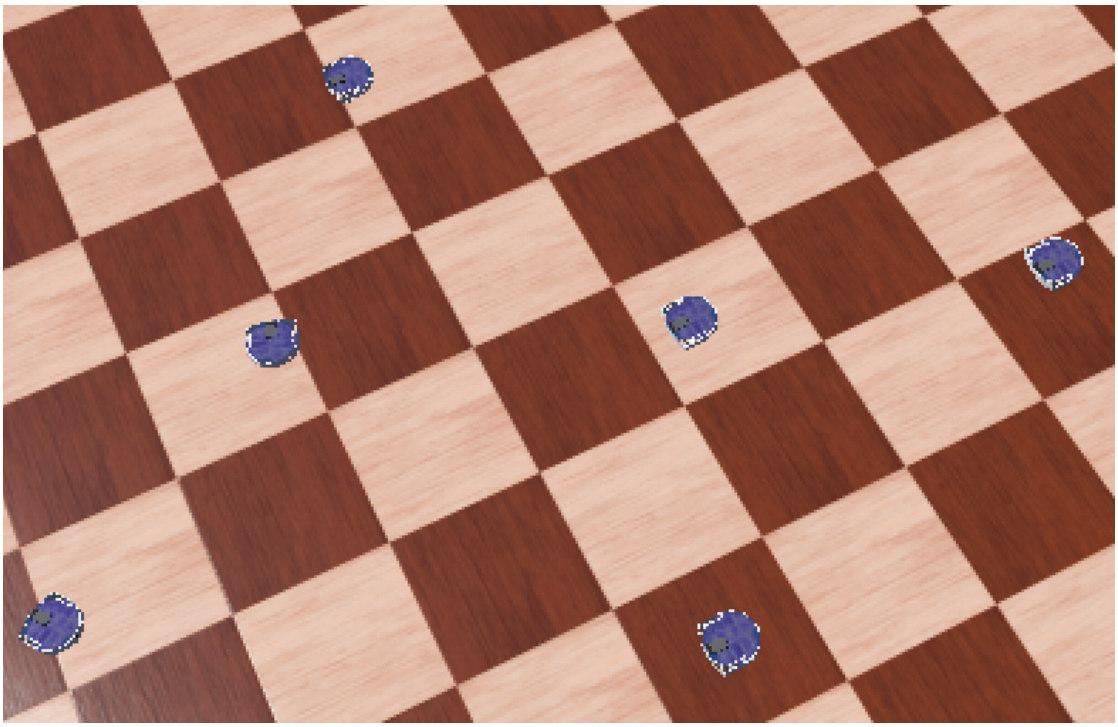


FIGURE 2.2 – Des robots Mona simulés avec WeBots

3 Localisation des robots

3.1 Rôle

La localisation permet de connaître la position et l'orientation de chaque robot. Ces informations permettent de communiquer aux robots la position et l'orientation de leurs voisins. Cela fonctionne comme un "super capteur".

3.2 Interférences OptiTrack × Mona

3.2.1 Le problème

Dans le stage précédant au mien, les robots Mona avaient été utilisés avec une table tangible qui utilise une caméra et un projecteur infrarouge pour détecter les objets posés sur elle. Il avait été constaté que les capteurs de proximité des Mona, qui utilise les infrarouges, étaient perturbés. Étant donné qu'OptiTrack utilise aussi des infrarouges, j'ai réalisé une expérience pour mettre en évidence une potentielle perturbation.

Cette expérience consiste à placer un robot en face d'un mur, de le faire avancer à vitesse constante et de relever les valeurs lues par les capteurs de proximités. Cette expérience a été réalisée en deux temps : une première fois avec les caméras d'OptiTrack éteintes, donc sans influence des projecteurs à infrarouge ; et une seconde fois avec les caméras allumées. L'expérience a ensuite été répétée avec différents robots.

Les graphiques de la figure 3.1 représentent les résultats de cette experience. Sur le premier graphique (3.1a), on constate que les capteurs 2 et 3 (capteurs frontaux) détectent la présence du mur au bout d'un certain temps (une valeur haute signifie un objet proche) et que les autres capteurs restent à un niveau bas, relativement constant. À l'inverse, sur le second graphique (3.1b), on constate que les capteurs 4 et 5 oscillent entre une valeur haute et une valeur basse très rapidement et que les autres capteurs ont de petits sur-sauts.

J'en conclus donc que les capteurs de proximité sont perturbés par les caméras d'OptiTrack.

Pour pallier à ce problème différentes solutions ont été envisagées et sont présentées ci-dessous dans la sous-section 3.2.2.

Un capteur de proximité à infrarouge est constitué de deux éléments : une led à infrarouge, qui peut émetre des la lumière infrarouge et d'une photodiode qui peut détecter la lumière infrarouge. La led émet de la lumère et la photodiode indique la quantité de lumière qui s'est réfléchit. Plus il y a de lumière qui se réfléchit et plus l'obstacle est proche.

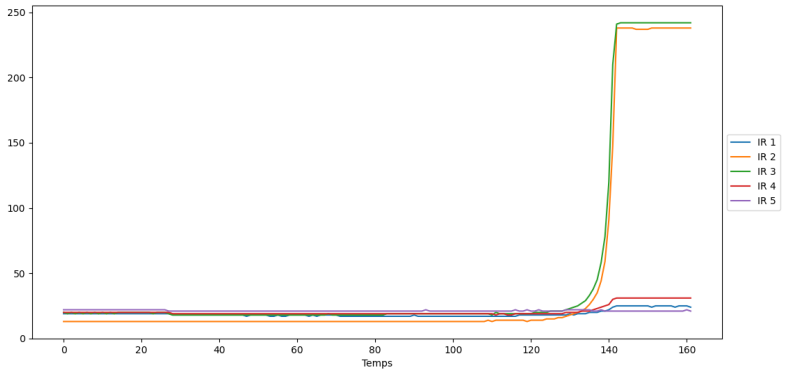
3.2.2 Les solutions

Solutions	Envisageable ?
Capteur ultra-sonique	✗
Changer de système de localisation	✓
Filtre optique	✓

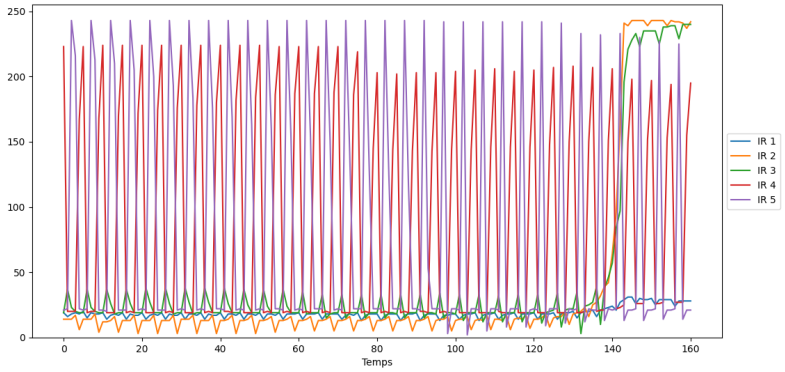
Table 3.1 – Récapitulatif des solutions proposées

3.2.2.1 Capteurs ultra-soniques

Les capteurs ultra-soniques fonctionnent sur le même principe que les capteurs à infrarouge mais avec des utrasons. Un émmetteur envoie une série d'ultrasons qui se réfléchissent sur les



(a) Caméras éteintes (pas d'influence)



(b) Caméras allumées

FIGURE 3.1 – Relevé des capteurs IR

FIGURE 3.2 – Exemple de marqueur utilisé par IRIDIA

obstacles, la durée entre l'émission et la réception indique la distance. Avec ces capteurs, il n'est pas possible de changer la fréquence des ultrasons utilisés. Dans le cas d'une utilisation avec un essaim de robots cela pose problème : les robots seraient incapables de distinguer les sons réfléchis par un obstacle et les sons émis par les autres robots. Cette solution n'est donc pas envisageable.

3.2.2.2 Utiliser un autre système de localisation

En amont de ce stage, j'ai réalisé un état de l'art de méthodes de localisation. J'ai réalisé une nouvelle phase de recherche pour trouver des solutions de localisation plus adapté à notre situation. Il existe différents moyens d'obtenir la localisation d'un objet, basé sur différents principes : optique (avec des caméras) ou encore radio-fréquence (en exploitant le *temps-de-vol*). Au vu des capacités des robots Mona, une solution basée sur de la reconnaissance d'image sera plus simple à mettre en place. Dans mon état de l'art, j'ai trouvé deux méthodes utilisant des caméras. La première, présentée dans Ribo et al. [7], est un dispositif de caméras stéréoscopiques permettant la localisation d'objet en trois dimensions. Néanmoins, cette technologie utilise des caméras à infrarouge, ce qui nous posera aussi problème avec les capteurs de proximité. La seconde technologie, IRIDIA, décrite dans Stranieri et al. [8], utilise une matrice de caméras disposées au plafond pour couvrir une zone. Grace à de la reconnaissance d'image, des marqueurs (figure 3.2) placés sur les robots sont reconnue et permet d'identifier chaque robot et de reconnaissance son orientation et sa position.

Après avoir contacté les auteurs de l'article, il s'avère que leur technologie repose sur une bibliothèque de traitement d'image propriétaire, ce qui la rend inutilisable pour notre projet. Cependant, l'un des auteurs m'a orienté vers un projet sur lequel il travaille : ARK (Augmented Reality for Kilobots) [6]. Ce dernier, repose sur le même principe : une matrice de caméras est placée au plafond pour couvrir une zone. Cependant, il est orienté pour être utilisé avec des Kilobots (un autre type de petit robot) et nécessiterait des modifications.

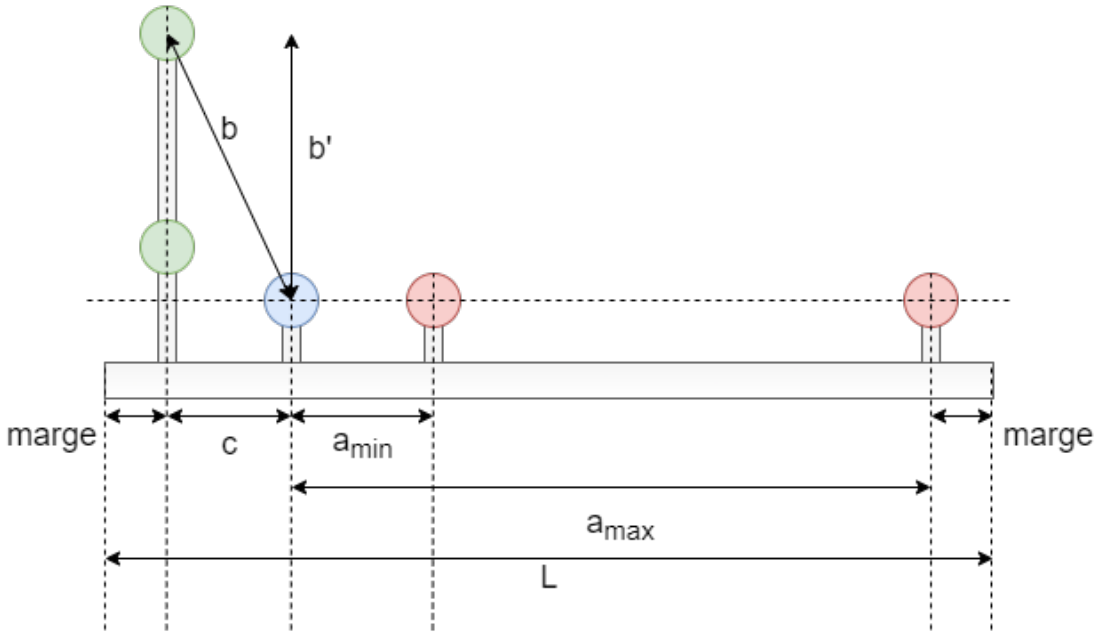
Changer de système de localisation à un coût en temps et en argent. Cette solution est donc gardée en dernier recours.

3.2.2.3 Filtre optique

La lumière infrarouge émise par les caméras d'OptiTrack est à une longueur d'onde de 850 nm. La led des capteurs de proximité émet à une longueur d'onde de 950 nm. On pourrait alors s'attendre à ce que le capteur de proximité soit sensible qu'à la longueur d'onde qu'il utilise. Cependant, les photodiodes sont généralement sensible à une bande de longueur d'onde étendue.

Heureusement pour nous, la longueur d'onde parasite et la longueur d'onde utile sont suffisamment éloignées l'une de l'autre pour pouvoir en filtrer la première en conservant la deuxième. Pour cela, nous allons utiliser un filtre passe-haut. Ce type de filtre laisse passer les longueurs d'onde supérieur à une valeur donnée, nommée longueur d'onde de coupure.

En modifiant un peu le casque des robots, on peut s'en servir comme support pour placer les filtres en face des capteurs.



Le marqueur bleu reste fixe, le rouge se déplace horizontalement et le vert verticalement.

FIGURE 3.3 – Position des marqueurs sur le robot

3.2.3 Validation de la solution

La solution retenue est celle des filtres optiques. Ces filtres peuvent être en verre ou en acrylique. J'ai opté pour l'acrylique, car celui-ci est souple et pourra épouser la courbure du casque. Sur le marché, il n'est pas possible de choisir la fréquence de coupure du filtre : les producteurs proposent certaines valeurs précises. J'ai choisi le modèle AC900 de MidOpt [1]. Ce modèle à une fréquence de coupure à 900 nm et bloque complétement la longueur d'onde parasite des caméras d'OptiTrack tout en laissant passer la lumière des capteurs de proximité. Les filtres sont vendus par plaque dont on peut choisir les dimensions. J'ai choisi de prendre une seule plaque suffisamment grande, par souci d'économie. Cette plaque sera ensuite découpée en petits filtres pour être placée sur les robots, avec l'aide du département optique d'IMT Atlantique.

Au moment où j'écris ces lignes, je n'ai pas reçu les filtres. Ils n'ont donc pas pu être testés en condition réelle.

3.3 Système de constellation

3.3.1 Pourquoi ?

Comme dit plus haut, OptiTrack reconnait les objets traqués en reconnaisant la position relative des différents marqueurs. L'utilisation d'un essaim de robots implique deux complications. La première est le nombre, chaque robot doit avoir au moins trois marqueurs (nombre minimal pour former un *rigid body*) et une constellation unique. La seconde est la taille, les robots étant petits, il y a peu d'espace pour placer les marqueurs. Il est donc important de créer un système pour générer un nombre suffisant de constellations différentiables.

3.3.2 Comment ?

Chaque constellation est composée de trois marqueurs (figure 3.3) : un qui reste fixe par rapport au robot et deux autres dont la position par rapport au premier marqueur changera. L'un de ces deux derniers marqueurs se déplacera sur le plan horizontal et l'autre verticalement pour gagner de la place sur le robot. Des tests avec OptiTrack ont permis de déterminer la distance minimale nécessaire entre deux marqueurs pour être différenciés (noté e_{min}) et aussi, la distance minimale dont il faut déplacer un marqueur pour la constellation soit considérée différente (noté

δ). À partir de ces valeurs on peut associer à chaque constallation un couple (n, m) de nombres entiers naturels. Ce couple code la position des deux marqueurs dont la position change.

Soit a_n, la distance entre le marqueur fixe et le marqueur sur le plan horizontal et soit b_m, la distance entre le marqueur fixe et le marqueur sur le plan vertical dans la constellation (n, m).

$$a_n = e_{min} + n.\delta \tag{3.1}$$
$$b_m = e_{min} + m.\delta \tag{3.2}$$

Afin de déterminer le nombre de combinaison à notre disposition, il nous faut déterminer la valeur maximum pour les distances a et b ; noté respectivement a_{max} et b_{max}.

Pour a_{max}, cela va dépendre du diamètre du robot (L), des marges $(margin)$ et de la distance c.

$$a_{max} = L - 2.margin - c \tag{3.3}$$

Pour b_{max}, cela va dépendre de l'encombrement verticale que l'on s'accorde. Notons b'_{max} la distance verticale entre le marqueur fixe et le marqueur du plan vertical.

$$b_{max} = \sqrt{{b'_{max}}^2 + c^2} \tag{3.4}$$

On peut alors calculer les valeurs miximales que peuvent prendre n et m.

$$n_{max} = \left\lfloor \frac{a_{max} - e_{min}}{\delta} \right\rfloor \tag{3.5}$$
$$m_{max} = \left\lfloor \frac{b_{max} - e_{min}}{\delta} \right\rfloor \tag{3.6}$$

Le nombre de combinaison est donc :

$$nb_{combinaison} = (n_{max} + 1)(m_{max} + 1) \tag{3.7}$$

On va à présent calculer l'encombrement vertical nécessaire pour obtenir un nombre de combinaisons voulues (noté nb_{wanted}).

$$\begin{aligned}
nb_{combinaison} &\geq nb_{wanted} \\
(n_{max} + 1).(m_{max} + 1) &\geq nb_{wanted} \\
m_{max} &\geq \frac{nb_{wanted}}{n_{max} + 1} - 1 \\
\left\lfloor \frac{b_{max} - b_{min}}{\delta} \right\rfloor &\geq \frac{nb_{wanted}}{n_{max} + 1} - 1 \\
\left\lfloor \frac{b_{max} - b_{min}}{\delta} \right\rfloor &\geq \left\lceil \frac{nb_{wanted}}{n_{max} + 1} - 1 \right\rceil \\
\frac{b_{max} - b_{min}}{\delta} &\geq \left\lceil \frac{nb_{wanted}}{n_{max} + 1} - 1 \right\rceil \\
b_{max} &\geq \left\lceil \frac{nb_{wanted}}{n_{max} + 1} - 1 \right\rceil \delta + b_{min}
\end{aligned} \tag{3.8}$$

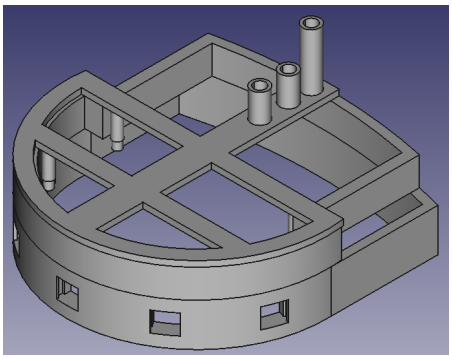
Il nous reste plus qu'a choisir certaines valeurs pour réaliser l'application numérique :

— $margin$: permet d'éviter que les marqueurs ne dépassent du bord du robot.

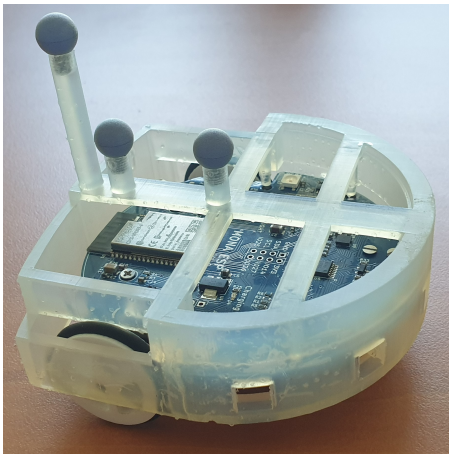
— c : suffissament petit pour gagner le plus possible de combinaisons, mais suffisament grand pour pouvoir placer le marqueur sans encombre.

— n_{wanted} : le nombre de robots devant être utilisés à terme.

L'application numérique a été réalisée avec Smath Studio, la feuille de calcule se trouve annexe A. Les tests pour valider ce système sont présentés section 5.1.



(a) Sous FreeCAD



(b) Robot avec le casque

FIGURE 3.4 – Casque V2

3.4 Casque V2

Lors d'un précédent projet, un casque pour les robots a été créé pour améliorer la détection des robots par les capteurs de proximité (figure 1.2b). Pour ce projet, les besoins ont évolué, nous avons maintenant les filtres optiques à disposer devant les capteurs et les supports pour les constellations. Ce casque à été créer avec FreeCAD, un logiciel de conception assistée par ordinateur (figure 3.4).

4 La passerelle

4.1 Rôle & Besoins

Les capacités sensorielles des robots sont limitées. Les algorithmes comportementaux nécessitent que les agents connaissent la position de leurs voisins. Les capteurs de proximité ne permettent pas de connaître la nature de l'obstacle (s'il s'agit d'un autre robot ou d'un mur) et ne perçoivent que vers l'avant.

Le rôle de la passerelle est justement d'agir, du point de vue des robots, comme un capteur, qui percevrait la présence des voisins et qui pourrait être artificiellement limité : en limitant, par exemple, la portée à 50 cm autour du robot. Pour ce faire, la passerelle recevra des données de localisation des robots depuis une source externe, qui peut être OptiTrack ou WeBots. Ces données sont ensuite traitées pour envoyer à chaque robot la position relative de ses voisins par rapport à lui.

De plus, cette passerelle sera réutilisée dans des projets futurs qui impliqueront l'utilisation d'autres éléments pouvant avoir besoin d'informations issues du système de tracking et/ou des robots. On peut imaginer, par exemple, l'utilisation d'un casque RA pour permettre à un utilisateur de visualiser dans l'environnement des informations en lien avec l'essaim de robots ou encore d'un interacteur tangible (ex : une manette) pour envoyer des ordres aux robots. La passerelle doit donc prendre en compte cette extensibilité dans son design. Pour cela, elle a été pensée sous forme de modules, chacun s'occupant d'une tâche bien précise.

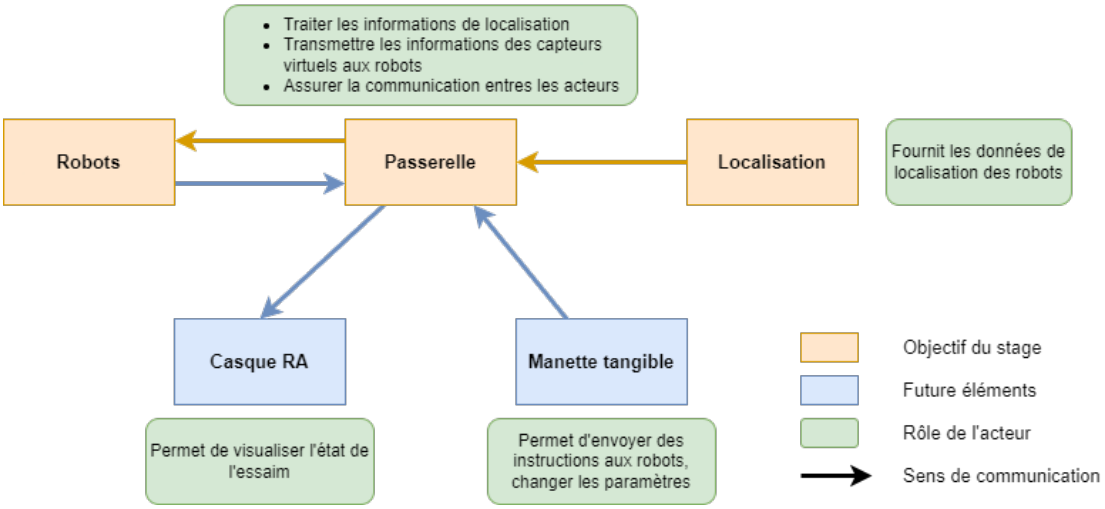


FIGURE 4.1 – Le projet dans sa globalité

4.2 Développement

4.2.1 Choix du language

Les langages hauts niveau, comme Python, ont l'avantage d'être facilement modulable et sont facile à maintenir. Néanmoins, il s'agit généralement de langages interprétés et accuse donc de lacunes en terme de performance. Étant donné que la passerelle aura à traiter plusieurs tâches

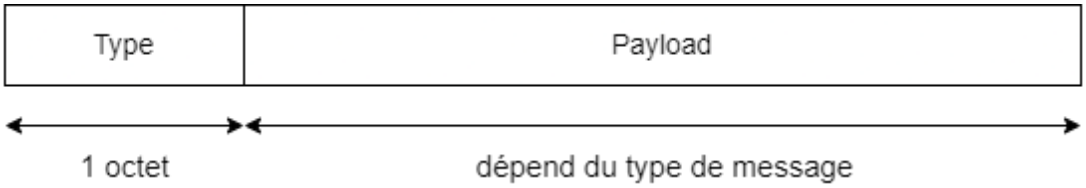


Figure 4.2 – Trame d'une commande

Nom	Description
Broadcast client tracking	Envoie un message à tous les clients de tracking
Enregistrement client de trakcing	Enregistre le client comme un client de trakcing

Table 4.1 – Commandes de la passerelle

en parallèle et que l'on souhaite réduire au minimum les latences, j'ai opté pour un langage bas niveau : Rust.

Il s'agit avant tout d'un choix personnel : j'ai une meilleure maîtrise du Rust et je trouve l'environnement de développement plus agréable. Il s'agit aussi d'un choix technique : Rust impose certaines règles strictes qui permettent de supprimer certaines classes de problème comme, par exemple, celles liées aux fuites de mémoire.

4.2.2 Choix du protocole de communication

Les robots Mona étant équipés d'un module Wi-Fi, la communication se fera via le protocole internet. Il existe deux protocoles pour envoyer des messages : le TCP et l'UDP. La différence est qu'avec TCP, les données sont garantie d'arriver à destination et dans le même ordre qu'ils ont été envoyés. Ceci est permis via un système d'acquittement lors de la réception. À l'inverse, l'UDP ne donne pas la garantie de la réception des paquets, mais à l'avantage de permettre le *broadcast* et le *multicast*. J'ai donc choisi l'UDP pour minimiser le traffic sur le réseau, mais aussi pour pouvoir utiliser le *broadcast* et le *multicast*.

Le protocole internet sera aussi utilisé pour la communication avec les autres acteurs. Il permet de communiquer facilement en local sur un même ordinateur entre deux logiciels. Par exemple entre WeBots et la passerelle. C'est aussi la méthode utilisée par OptiTrack pour streamer les informations de localisation.

Le format des messages en eux même, dépend de l'acteur avec qui la passerelle communique. Ces formats sont décrit ci-dessous dans les sous-sections correspondantes de la sous-section 4.2.3.

4.2.3 Les modules

4.2.3.1 Serveur de commande

Le serveur de commande sert à traiter des requêtes provenants des systèmes externes. Les commandes sont reçues via un socket UDP et dont la trame doit correspondre à la figure 4.2.

Les différentes commandes sont décrites dans le tableau 4.1

4.2.3.2 Distributeur de message

Ce module a pour rôle d'envoyer des messages aux acteurs via un socket UDP. Il reçoit des requêtes d'envoi de la part des autres modules.

4.2.3.3 Données de tracking

Ce module a pour rôle de recevoir les données de tracking d'une source externe et de convertir les données pour correspondre à ce qu'attends les robots. Ce module possède deux modes de fonctionnement : direct et OptiTrack.

Le mode direct s'attend à recevoir la position sur le plan du robot et sa rotation.

Le mode OptiTrack permet de recevoir des données directement depuis Motive (le logiciel utilisé par OptiTrack) et en extrait la position du robot dans le plan et sa rotation. OptiTrack

communique avec le protocole NatNet. Un client NatNet partiel a été recréer pour pouvoir lire les données ; seules les parties du protocole qui nous intéresse on été recréer.

4.2.3.4 Envoie des données de tracking

Ce module utilise les données de tracking formatées par le module précédant, puis les envoie à chaque client de tracking à tour de rôle suivant le principe du *round-robin*. Les clients sont inscrits dans une liste, on sert le premier client, puis on passe au suivant, et ainsi de suite. Si un nouveau client s'inscrit en cours de route, on lui envoie les données dont il a besoin immédiatement, puis on le place dans la liste juste avant le client en cours. De cette manière, on permet au nouveau client de recevoir rapidement sa première mise à jour et en même temps, il devra attendre un cycle complet pour recevoir la prochaine. On garantie, ainsi une certaine équité.

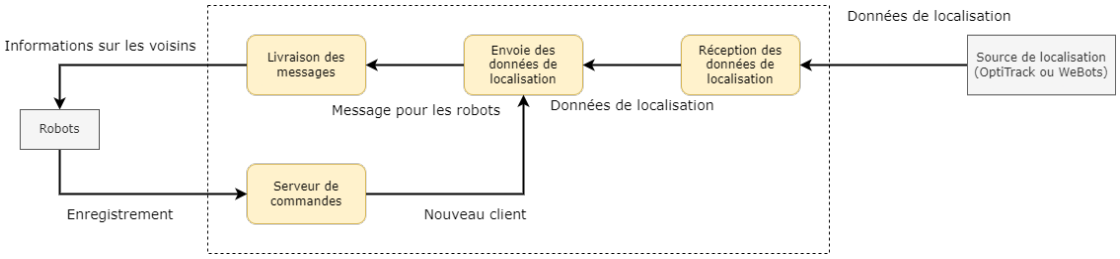


FIGURE 4.3 – Fonctionnement interne de la passerelle

4.3 Paramétrable

Pour rendre son fonctionnement plus adaptable, la passerelle est paramétrable. On peut régler quels sont les voisins qui seront détectés par un robot : tous les voisins, les voisins dans un rayon spécifique, les n plus proches voisins ou encore les n plus proches voisins dans un rayon donné. Il est aussi possible de régler le délai entre deux envoie d'informations à un robot.

5 Validation

Dans cette partie, je présente les tests qui ont été réalisés afin de valider le fonctionnement du tracking des robots et de la passerelle.

5.1 Tracking des robots

Afin de tester le système de constellation, des supports ont été imprimés, sur lesquels des marqueurs d'OptiTrack ont été fixés. Des constellations avec des combinaisons proches ($(0,0)$, $(0,1)$, $(1,0)$ et $(7,4)$, $(7,3)$, $(6,4)$) ont été utilisées. Je souhaite ici vérifier qu'OptiTrack arrive à correctement labelliser les constellations. Pour tester cela, j'ai déplacé les constellations en les plaçant sur une table, pour simuler les mouvements des robots. Il en ressort que certaines constellations sont mal reconnues. Les distances a et b (sous-section 3.3.2) sont calculées avec des formules identiques, avec a basée sur n et b basée sur m. Il en résulte que les constellations de la forme (n,n) présentent un axe de symétrie, OptiTrack ne peut pas savoir si le robot est à l'horizontale ou à la verticale. Les constellations de la forme (n,m) et (m,n) sont symétriques l'une par rapport à l'autre, OptiTrack peut les confondre si on les utilise simultanément. Étant donné que n peut prendre des valeurs de 0 à 4, on perd, en réalité, que 9 combinaisons : de $(0,0)$ à $(4-4)$ ainsi que $(0,1)$, $(1-2)$, $(2-3)$ et $(3-4)$. On peut compenser cela facilement en augmentant le nombre sur m. Je considère donc que le système de constellations est validé.

5.2 La passerelle

5.2.1 Tests unitaires

Pour valider le fonctionnement de la passerelle, j'ai, dans un premier temps, identifié les fonctions unitaires pour les tester une à une. Ces fonctions sont les suivantes :

- Récupérer les trois degrés de libertés qui nous intéressent depuis OptiTrack.
- Obtenir directement les trois degrés de libertés depuis n'importe quel source (par exemple WeBots)
- Établir la communication avec un robot physique à travers un routeur
- Établir la communication avec un robot virtuel (sous WeBots) en locale

Dans le cas de la communication avec un robot physique ou virtuel, la partie tracking est émulée à l'aide d'un script.

Ces points fonctionnent correctement.

5.2.2 Fonctionnement Global

Dans un second temps, j'ai testé l'ensemble du système dans deux cas de figure : dans le cadre d'une simulation sous WeBots et en physique avec OptiTrack. Pour valider que les informations transmises par la passerelle sont correctes, j'ai réalisé l'expérience suivante : on utilise deux robots, d'un inactif qui sert de cible et d'un autre dont l'objectif est de se rendre sur son voisin. Étant donné qu'il n'y a qu'un voisin, il n'y a pas d'ambiguïté. Le robot avance jusqu'à qu'il se trouve à moins de 10 cm de sa cible. Pour visualiser l'état interne du robot, je me sers des leds avec un code couleur : rouge s'il avance vers son voisin, bleu s'il a atteint sa cible et orange s'il ne détecte pas de voisin. Si le robot avance vers ça cible et s'arrête lorsqu'il l'atteint, on pourra dire que les informations transmises par la passerelle sont correctes. Ces tests ont permis de mettre en lumière un problème avec les systèmes de coordonnées : WeBots utilise un repère avec l'axe Z pour le haut et OptiTrack utilise l'axe Y pour le haut. Ce problème a pu être corrigé et les informations transmises par la passerelle validées.

6 Conclusion

6.1 Avancement & Continuïté

Le tracking des robots ainsi que la passerelle ont pu être validés. La validation des filtres optique n'a pas pu se faire faute de temps.

Il me reste un mois de stage qui sera consacré à la production des documents techniques des différents éléments que j'ai développés. Et si je reçois les filtres au cours de ce mois, je validerai leur fonctionnement en réalisant une expérience similaire à celle qui m'a permis de mettre en évidence l'interférence des caméras d'OptiTrack : un robot équipé des filtres avance à vitesse constante vers un mur, on relève les valeurs retournées par les capteurs de proximité. On réalise cette expérience avec et sans les caméras d'allumer.

6.2 Bilan

De la conception 3D, des algorithmes comportementaux, de la communication en réseau, du développement de logicielle, ce stage a été une expérience pluridisciplinaire qui m'a permis d'exploiter et d'étendre mes capacités.

J'ai malheureusement perdu beaucoup de temps à découvrir, comprendre et résoudre le problème de conflis entre le système de tracking et les capteurs de proximité des robots.

Bibliographie

[1] *AC900 Acrylic SWIR Longpass | Custom Laser Cut | Covert Surveillance*. URL : https://midopt.com/filters/ac900/ (visité le 13/06/2022).

[2] Farshad Arvin et al. "Mona : an Affordable Open-Source Mobile Robot for Education and Research". In : *Journal of Intelligent & Robotic Systems* 94.3 (juin 2019), p. 761-775. ISSN : 0921-0296, 1573-0409. DOI : 10.1007/s10846-018-0866-9. URL : http://link.springer.com/10.1007/s10846-018-0866-9 (visité le 13/06/2022).

[3] *Interaction*. URL : https://labsticc.fr/fr/poles/interaction (visité le 20/06/2022).

[4] *Laboratoire des sciences et techniques de l'information, de la communication et de la connaissance*. In : *Wikipédia*. Page Version ID : 185518827. 15 août 2021. URL : https://fr.wikipedia.org/w/index.php?title=Laboratoire_des_sciences_et_techniques_de_l%27information,_de_la_communication_et_de_la_connaissance&oldid=185518827 (visité le 20/06/2022).

[5] *Présentation*. URL : https://labsticc.fr/fr/lab-sticc/presentation (visité le 20/06/2022).

[6] Andreagiovanni Reina et al. "ARK : Augmented Reality for Kilobots". In : *IEEE Robotics and Automation Letters* 2.3 (juill. 2017). Conference Name : IEEE Robotics and Automation Letters, p. 1755-1761. ISSN : 2377-3766. DOI : 10.1109/LRA.2017.2700059.

[7] M. Ribo, A. Pinz et A.L. Fuhrmann. "A new optical tracking system for virtual and augmented reality applications". In : *IMTC 2001. Proceedings of the 18th IEEE Instrumentation and Measurement Technology Conference. Rediscovering Measurement in the Age of Informatics (Cat. No.01CH 37188)*. IMTC 2001. Proceedings of the 18th IEEE Instrumentation and Measurement Technology Conference. Rediscovering Measurement in the Age of Informatics (Cat. No.01CH 37188). T. 3. ISSN : 1091-5281. Mai 2001, 1932-1936 vol.3. DOI : 10.1109/IMTC.2001.929537.

[8] Stranieri, A et al. "IRIDIA's arena tracking system". In : *IRIDIA, Université Libre de Bruxelles*. 2013.

[9] *Webots*. In : *Wikipedia*. Page Version ID : 1061385751. 21 déc. 2021. URL : https://en.wikipedia.org/w/index.php?title=Webots&oldid=1061385751 (visité le 13/06/2022).

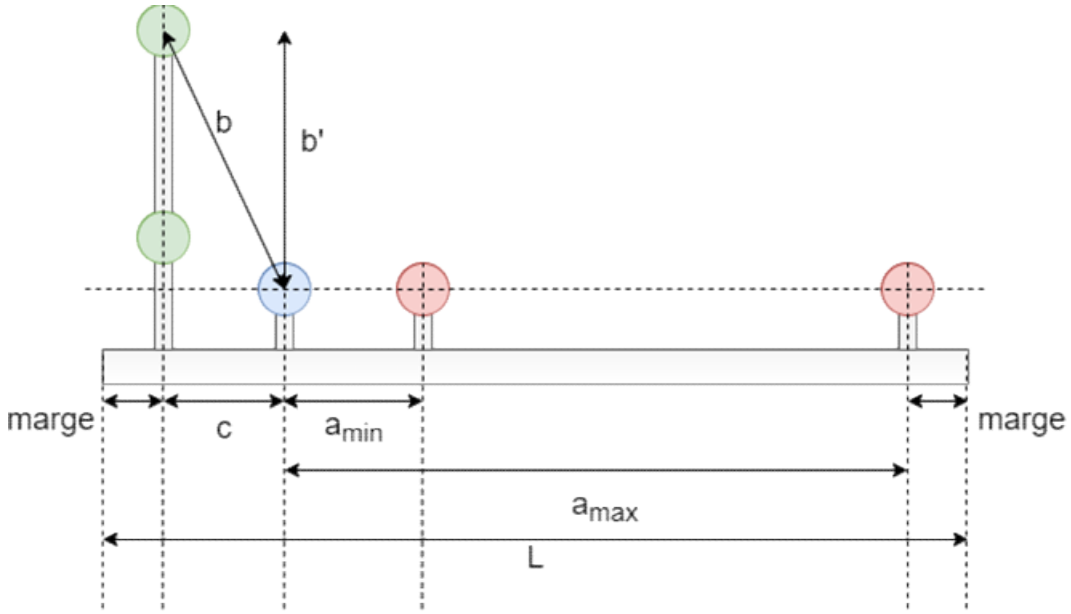
A Feuille de calcules pour le système de constellation

Légende

Données

Valeurs choisies

Résultats



$ecart\_min := 2{,}7 \cdot D_{markeur}$ Écart minimal entre deux marqueurs pour les différencier (valeur empirique)

On définit la disance a et b en fonction des positions n et m (nombres entiers).

$a(n) := a_0 + n \cdot \delta \qquad a_0 := ecart\_min$

$b(m) := b_0 + m \cdot \delta \qquad b_0 := ecart\_min$

$\delta := \frac{D_{markeur}}{2}$ Écart minimal pour différencier deux constellations (valeur empirique)

On peut alors calculer les valeurs maximum que peuvent prendre n et m :

$n_{max} := floor\left(\frac{a_{max} - a_0}{\delta}\right) \qquad m_{max} := floor\left(\frac{b_{max} - b_0}{\delta}\right)$

$nb_{combinaison} := (n_{max} + 1) \cdot (m_{max} + 1)$

$a_{max} := L - 2 \cdot margin - c$ Valeur maximal de la distance a

$b_{max} := \sqrt{b'_{max}{}^2 + c^2}$ Valeur maximal de la distance b

$b'_0 := \sqrt{b_0{}^2 - c^2}$ Valeur minimal de la distance b'

On peut calculer l'encombrement vertical minimal pour obtenir un nombre de combinaison voulue

$$nb_{combinaison} \geq nb_{wanted}$$

$$(n_{max}+1).(m_{max}+1) \geq nb_{wanted}$$

$$m_{max} \geq \frac{nb_{wanted}}{n_{max}+1} - 1$$

$$\left\lfloor \frac{b_{max}-b_{min}}{\delta} \right\rfloor \geq \frac{nb_{wanted}}{n_{max}+1} - 1$$

$$\left\lfloor \frac{b_{max}-b_{min}}{\delta} \right\rfloor \geq \left\lceil \frac{nb_{wanted}}{n_{max}+1} - 1 \right\rceil$$

$$\frac{b_{max}-b_{min}}{\delta} \geq \left\lceil \frac{nb_{wanted}}{n_{max}+1} - 1 \right\rceil$$

$$b_{max} \geq \left\lceil \frac{nb_{wanted}}{n_{max}+1} - 1 \right\rceil \delta + b_{min}$$

Valeur minimal que doit prendre b max

$$b_{max,min} := ceil\left(\frac{nb_{wanted}}{n_{max}+1} - 1\right) \cdot \delta + b_0$$

Valeur minimal que doit prendre b'max

$$b'_{max,min} := \sqrt{b_{max,min}^{\,2} - c^2}$$

Données

$L := 8\ \text{cm}$ Diamètre du mona

$D_{markeur} := 9,5\ \text{mm}$ Diamètre des marqueurs

Valeurs fixées

$nb_{wanted} := 40$ Nombre de combinaisons voulues

$margin := \frac{D_{markeur}}{2} = 4,75\ \text{mm}$

$c := 9\ \text{mm}$

Résultats

$a_0 = 25,65\ \text{mm}$ $b'_0 = 24,0192\ \text{mm}$

On arrondie b'0 à deux decimales

$b'_0 := \text{round}\left(\frac{b'_0}{\text{mm}}; 2\right)\ \text{mm} = 24,02\ \text{mm}$

On re-calcule la valeur de b0

$b_0 := \sqrt{b'_0{}^2 + c^2} = 25,6507\ \text{mm}$

Valeur minimale que doit prendre b'max

$b'_{max,min} = 43,7343\ \text{mm}$

On fixe b'max

$b'_{max} := 43,8\ \text{mm}$

Nombre de combinaisons obtenues

$nb_{combinaison} = 40$

$n_{max} = 7$

$m_{max} = 4$