UNIVERSITÉ
CAEN
NORMANDIE

LOISEL Lucas [21711075]
GUERET Morgan [21700094]

MASTER 1 INFORMATIQUE

# Rapport : Intelligence artificielle distribuée

## Table des matières



Contrôle Continu

*BONNET Grégory et VALLEE Thibault*

Réalisé durant l'année de M1 Informatique 2020/2021

# 1 Introduction

SoccerBots est un jeu de foot tiré de la distribution TeamBots. La distribution est composée de deux applications Java et de plusieurs packages Java utilisés pour la recherche en robotique mobile.

Le jeu de foot, SoccerBots, quant à lui, reprend les principes élémentaires du sport dont il est tiré et donc chaque équipe doit marquer dans le camp adverse pour espérer l'emporter. Néanmoins, les 2 équipes sont composées de seulement 5 joueurs (à contrario des 11 joueurs habituels). De plus, nous ne retrouvons pas les règles d'arbitrage conventionnel du foot.

Ainsi, nous avons eu pour projet de développer une intelligence artificielle permettant de contrôler une équipe SoccerBots la plus performante possible.

Tout au long de ce rapport, nous allons exposer et expliquer la stratégie que nous avons suivie pour obtenir une IA la plus élaborée possible. Pour ce faire, nous commencerons par présenter la stratégie globale pour notre équipe (à travers les rôles, règles et relations d'inhibitions inculquées aux robots).

Ensuite, nous allons étudier le fonctionnement de la simulation (à travers les vecteurs et donc à travers la représentation des déplacements). Pour finir, nous allons tester notre équipe contre BasicTeam et GoToBall. Nous allons de plus faire affronter notre équipe (monojoueur) contre BasicTeam et GoToBall.

Par rapport à l'organisation de notre projet, nous avons séparé les fonctions jugées mathématiques et indépendantes de la classe ControlSystemSS dans un paquetage nommé TeamLoiselGueretPackage. Ce paquetage est à mettre dans TeamBots/src/.

## 2 Stratégie de notre équipe

En ce qui concerne notre stratégie, nous avons d'abord élaboré des rôles prédéfinis pour chacun de nos joueurs à travers trois méthodes (goalIsBetter, defenserIsBetter et strikerIsBetter). Ainsi, le joueur numéro 0 est notre goal, les joueurs numéro 3 et 1 jouent le rôle de défenseurs, mais non pas pour autant exactement les mêmes règles (un de nos défenseurs est plus offensif que l'autre). Finalement, les joueurs numéro 2 et 4 sont nos attaquants qui comme les défenseurs ne possèdent pas exactement les mêmes règles.

```
/**
 * Retourne le vecteur représentant l'action que le goal doit effectuer
 * @return un vecteur
 */
private Vec2 goalIsBetter()
{...}


/**
 * Retourne le vecteur représentant l'action que le défenseur doit effectuer
 * @return un vecteur
 */
private Vec2 defenderIsBetter()
{...}


/**
 * Retourne le vecteur représentant l'action que l'attaquant doit effectuer
 * @return un vecteur
 */
private Vec2 strikerIsBetter()
{...}
```

FIGURE 1 – Implémentation générale

Plus précisément, notre goal se place proche du but et reste entre le but et le ballon(dans un certain rayon par rapport au but, autrement dit, la norme maximale entre le vecteur et le goal est fixée pour définir "la zone" du goal) grâce à notre précieuse méthode betweenBallFromOurgoal.

```
Vec2 result = new Vec2(ourgoal);
abstract_robot.setDisplayString("to ourgoal");

if (ball.r < 0.6 && ourgoal.r < 0.6)
{
    result = betweenBallFromOurgoal();
    abstract_robot.setDisplayString("between ball");
```

FIGURE 2 – Fonctionnement du goal (dans la méthode goalIsBetter)

Pour nos défenseurs, le numéro 1 va vers la balle si elle se trouve dans un rayon de 1.4 par rapport à notre but. Sinon, nous considérons que nous sommes en "phase" offensive. Le robot se place ainsi au centre pour être le premier sur la balle lors d'une réapparition de cette dernière. Quant à lui, le robot numéro 3 se positionne dans un certain rayon par rapport à notre but et se met en attente. Cependant, le robot peut se déplacer dans un rayon de 1.4 s'il est le plus proche de la balle. Nous pouvons considérer ce robot comme plus défensif/en retrait.

```
if (mynumber == 1)
{
    if(toward(ball, ourgoal).r < 1.4) // va vers la balle si elle se trouve dans un rayon de 1.4 par rapport au but
    {...}
    else // Sinon va au centre pour attendre la balle durant la phase offensive
    {...}
}
else if (mynumber == 3)
{
    if(isClosestToBall() || toward(ball, ourgoal).r < 1.4){...}// si il est le plus proche de la balle alors ce place entre  leur adverse et la ball dans un rayon de 1.4 par rapport au but
    else // Sinon se place dans un défense dans un rayon compris entre 0.3 et 0.6 par rapport au but
    {...}
}
```

FIGURE 3 – Fonctionnement bref des défenseurs

Nos attaquants ont un rôle symétrique opposé aux défenseurs et donc le numéro 2 se place derrière la balle vers le but adverse si la balle se trouve dans un rayon supérieur à 1.2 par rapport au but. Sinon, nous considérons que nous sommes en phase défensive. Et donc, le robot se place au centre pour être le premier sur la balle lors d'une réapparition de cette dernière. Quant à lui, le numéro 4 se situe dans un certain rayon par rapport au but adverse. Cependant, le robot se place derrière la balle vers le but adverse s'il est le plus proche de la balle que ses alliés. Il peut aussi bloquer un joueur ennemi si un allié monte au but pour marquer. Nous pouvons considérer ce robot comme plus offensif et bloqueur.

```
if (mynumber == 2)
{
    if(toward(ball,ourgoal).r > 1.2) // se place derriere la balle s'il se trouve dans un rayon supérieur à 1.2 par rapport à notre but
    {...}
    else  // Sinon va au centre pour attendre la balle
    {...}
}
else if (mynumber == 4)
{
    if (isClosestToBallFromTeammates())
    {...} // se place derrière la balle vers le but adverse si plus proche de la balle
    else
    {...} // Sinon se place en attaque dans un rayon compris entre 0.3 et 0.6 par rapport à leur but
}
```

FIGURE 4 – Fonctionnement bref des attaquants

Ainsi, nous séparons le fonctionnement (les règles et relations d'inhibitions) de nos joueurs en fonction de la position qu'ils soient, goal, défenseurs ou attaquants. Cependant, comme nous l'avons évoqué précédemment et comme il est possible de le voir sur les images non-exhaustives (pour éviter d'être barbant) tirées de notre code, nous retrouvons quelques nuances en fonction des robots possédant une même position.

Dans la partie suivante, nous allons évoquer à la fois le plus intéressant et le plus technique (compliqué)

## 3 Fonctionnement de la Simulation

La simulation se déroule sur un terrain avec une échelle plus petite que le terrain du sport. Nous avons donc un terrain de 152.5 cm par 274 cm, des buts de 50 cm de large, une balle de 40 mm de diamètre et des robots de 12 cm de diamètre.

Nous pouvons sélectionner les équipes grâce au fichier robocup.dsc, il contient différentes variables comme la création du terrain, de la balle ou bien la durée de la partie. Cependant, ce qui nous intéresse est la création des équipes. Nous pouvons remarquer qu'il y a deux camps ("west" et "east") qui correspondent aux côtés du terrain. La simulation possède une position par défaut (schéma ci-dessous) avec l'équipe "west" qui effectue l'engagement :

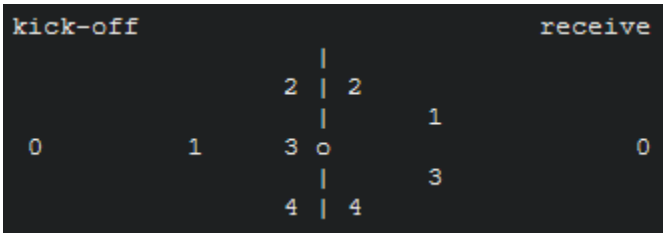



Figure 5 – Position initiale de la simulation

Nous observons des numéros, ces numéros sont attribués aux joueurs à l'initialisation de la simulation, chaque joueur conserve son numéro, il est ainsi possible de leur attribuer une action selon leur numéro.

Chaque objet est représenté par un vecteur de déplacement au début du tour de jeu d'un joueur avec deux cas différents :

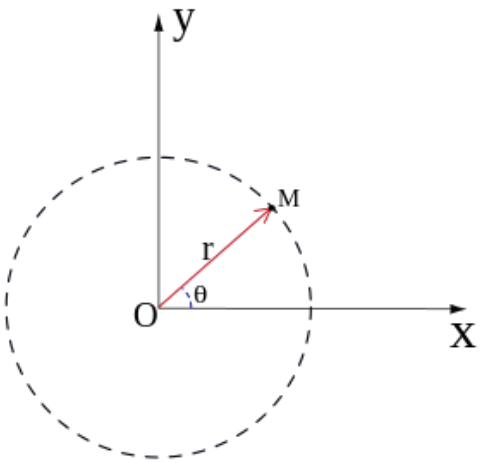
- Le joueur actuel est représenté par un vecteur du centre du terrain vers lui-même.
- Les autres objets sont représentés par un vecteur du joueur actuel vers eux.

A noter que les joueurs ne se déplacent pas pour s'arrêter à une coordonnée précise mais se déplacent selon un vecteur $(x, y)$.

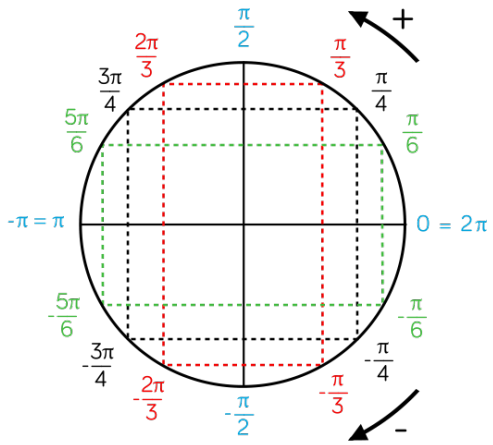
Les objets sont donc représentés à partir d'un point. En partant du principe que par rapport au centre, le côté "ouest" est représenté par un vecteur avec un $x < 0$. Tandis que, le côté "est" est représenté par un vecteur avec un $x > 0$.

Grâce à l'addition/soustraction de vecteurs, nous pouvons changer le point initial du vecteur, par exemple avoir la position de la balle non plus par rapport au joueur actuel, mais par rapport au centre. Chaque vecteur possède 4 variables importantes à prendre en compte $(x, y, r, t)$ :

- $x$ : définit le déplacement sur l'axe des abscisses du vecteur ($x = -1$ : vers l'ouest, $x = 1$ : vers l'est)
- $y$ : définit le déplacement sur l'axe des ordonnées du vecteur ($y = -1$ : vers le sud, $x = 1$ : vers le nord)
- $r$ : représente la distance du centre vers le point de coordonné $(x, y)$, c'est ce qu'on appelle la coordonnée radiale
- $t$ ou $\theta$ : représente la coordonnée angulaire entre $r$ et la demi-droite d'angle $0\pi$ allant de 0 à $2 * \pi$ ou $2 * -\pi$ ($t = 0$ : vers l'est, $t = \pi$ : vers l'ouest)




(a) Représentation d'un vecteur.




(b) Mesure d'un angle en radian.

Les schémas ci-dessus représentent respectivement un vecteur et ses variables $(x, y, r, \theta)$ ainsi que les valeurs possibles de l'angle $\theta$. En prenant en compte le fonctionnement des vecteurs nous pouvons réussir à calculer des déplacements précis vers une destination. Il est possible de positionner un joueur à un endroit précis en jouant sur la vitesse.

Nous utilisons principalement le $r$ des vecteurs pour déterminer la distance. Il permet de déterminer la proximité entre des objets (joueurs, balle, but). Et donc, de définir par exemple le joueur le plus proche de la balle ou encore la distance par rapport au but...

La variable $t$ des vecteurs est utilisée pour le positionnement d'un joueur par rapport à des objets. Nous pouvons grâce à elle déterminer si un joueur est positionné derrière la balle par rapport à un but ou même entre la balle et un but grâce au principe du schéma ci-dessous.

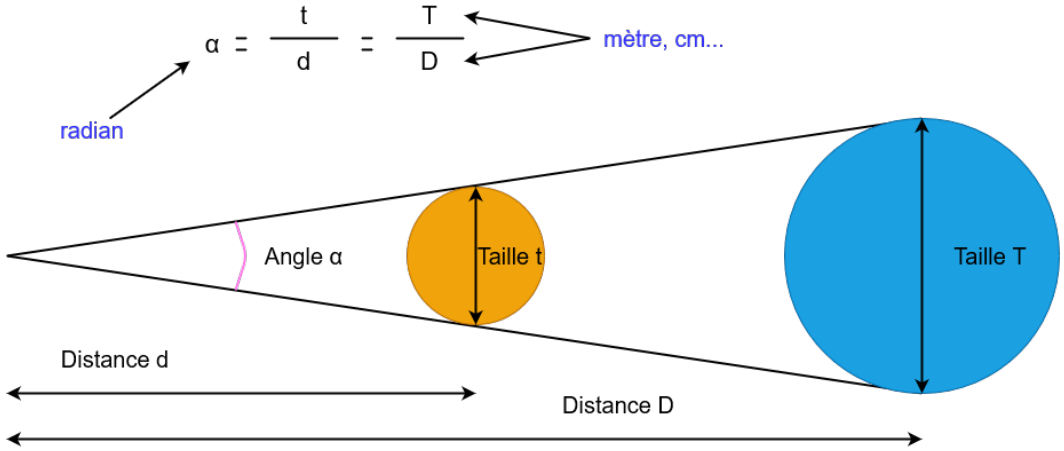



FIGURE 7 – Calcul d'un angle selon la distance et la taille d'un objet

Le schéma définit comment, en partant d'un point nous pouvons connaître l'angle par rapport à une destination en utilisant la taille de la destination et la distance entre notre point et le milieu de la destination. Sur ce principe, nous pouvons comparer si l'angle d'un objet est compris dans celui d'un autre objet par rapport à une destination et ainsi savoir s'il est dans son axe.

Pour pouvoir nous positionner derrière ou entre une cible et une destination, nous utilisons un vecteur de la destination vers la cible. Grâce à ce vecteur, nous avons une ligne d'endroit possible pour se positionner correctement derrière ou entre.

Pour éviter d'aller vers la balle, si nous souhaitons nous placer derrière la balle, nous ajoutons au $r$ la taille du joueur et la taille de la balle sinon nous retirons la taille du joueur et la taille de la balle (voir le schéma ci-dessous). Une fois notre position déterminée, nous remettons le vecteur dans un plan par rapport au joueur.

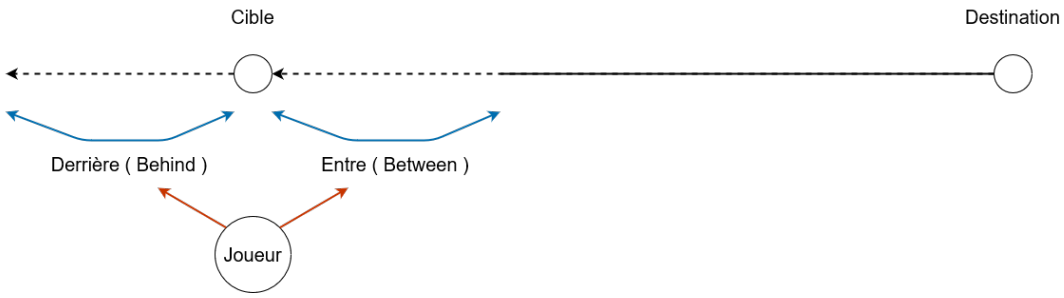



FIGURE 8 – Position derrière/entre

Nous avons aussi essayé de placer selon l'angle et la distance pour pouvoir prendre une courbure plus large, mais cette méthode s'avérait être moins efficace, car elle laissait trop d'espace à l'adversaire.

# 4 Tests contre BasicTeam et GoToBall

Pour réaliser les tests, nous avons décidé de faire des matchs de 1 minute. Nous avons effectué 1 match à l'ouest et un match à l'est contre chaque équipe hormis pour notre équipe monojoueur contre GoToBall qui ne joue qu'à l'est.

Grâce aux tests, nous pouvons dire que notre équipe est performante contre des équipes comme BasicTeam et GoToBall. Il y a eu quelques problèmes de position bloquante contre GoToBall.

A travers nos tests, nous pouvons remarquer que l'équipe complète n'a pas pris de but tandis que l'équipe mono joueur a été plus efficace contre BasicTeam.

## 4.1 Tests équipe complète

Equipe complète jouant à l'ouest :

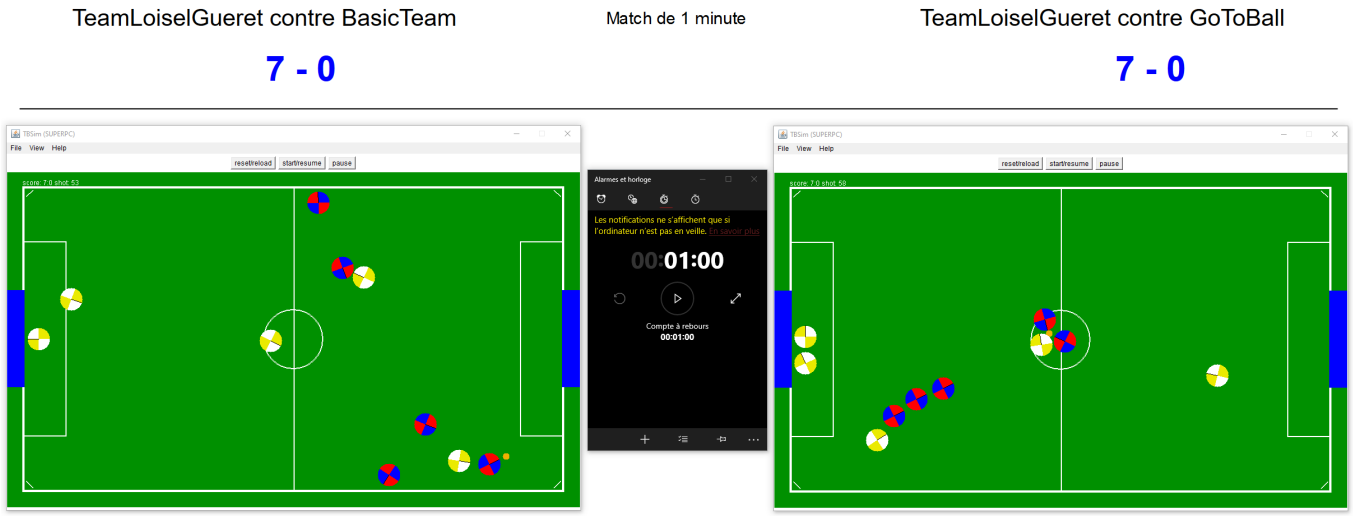



FIGURE 9 – Premier test, avec pour score 7-0 et 7-0 pour notre équipe sur 1 minute en étant à l'ouest.

Notre équipe complète a remporté les deux matchs à l'ouest sans encaisser de but.

Equipe complète jouant à l'est :

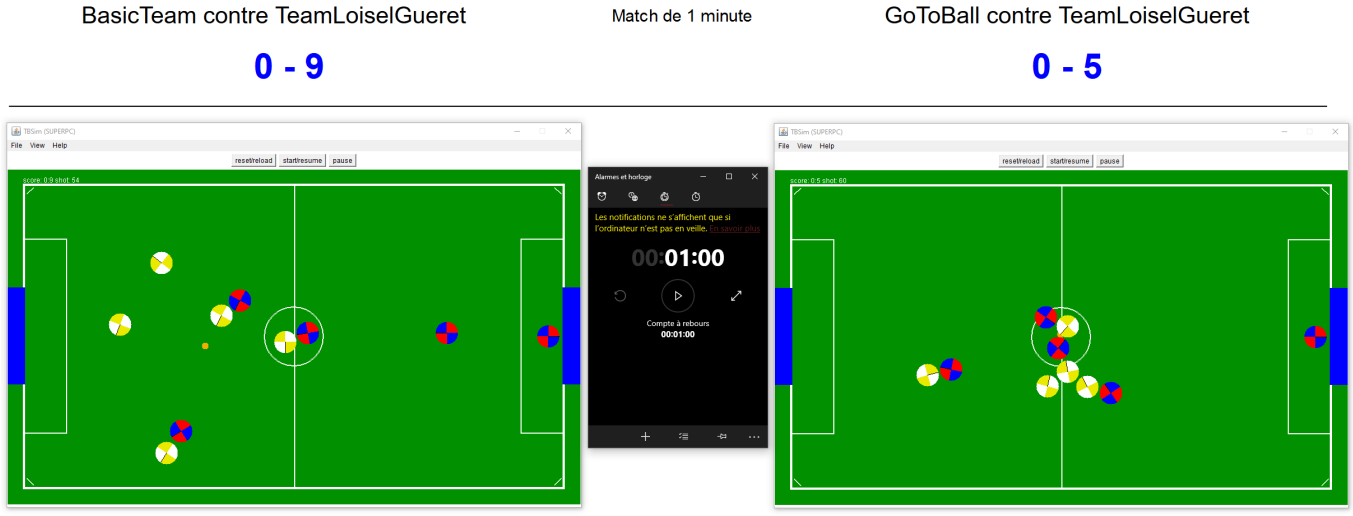



FIGURE 10 – Second test, avec pour score 0-9 et 0-5 pour notre équipe sur 1 minute en étant à l'est.

Notre équipe complète a remporté les deux matchs à l'est sans encaisser de but. Cependant, le match contre GoToBall a fini dans une position bloquante.

## 4.2 Tests équipe monojoueur

Equipe mono joueur jouant à l'ouest contre BasicTeam, à l'est contre GoToBall :


FIGURE 11 – Premier test, avec pour score 9-1 et 1-6 pour notre équipe (mono joueur) sur 1 minute.

Notre équipe mono joueur a remporté ses deux matchs, mais pour les deux cas nous avons encaissé 1 but(pour GoToBall, notre équipe mono joueur joue à l'est, car sinon, la balle reste bloquée au centre, bloquée par les joueurs).

Equipe mono joueur jouant à l'est :


FIGURE 12 – Second test, avec pour score 0-14 et 1-5 pour notre équipe (monojoueur) sur 1 minute.

Notre équipe mono joueur a remporté ses deux matchs, mais en encaissant 1 but contre GoToBall.

# 5 Conclusion

Pour conclure, ce rapport, nous avons avancé différents points clé de notre projet. D'abord avec la stratégie globale de notre équipe qui reprend les grandes lignes (idées) du fonctionnement de notre équipe.

Ensuite, nous avons étudié les vecteurs et leur fonctionnement qui est la partie la plus importante pour diriger convenablement nos robots.

Pour finir, nous avons exposé quelques tests en faisant affronter nos équipes contre Basic Team et GoToBall.

Nous avons vraiment apprécié faire ce projet malgré les quelques difficultés rencontrées sur le fonctionnement des vecteurs. De plus, nous n'avons malheureusement pas pu faire tout ce que nous voulions implémenter.

Effectivement, il serait intéressant plus tard d'ajouter un système de passe élaboré entre les joueurs afin d'avoir une équipe plus performante.