INSA LYON

SCOR – Rapport Technique

Equipe : Adrien BROCHOT – Armand ROSSIUS – Jetmir XHEMBULLA
28/02/2012

Table des matières

I. Présentation du projet

1. Le projet

Le projet SCOR a pour objectif de faire s'affronter deux équipes de robots dans un match assimilable à du football simplifié. Le projet existe maintenant depuis 4 ans et 4 équipes ont pu être constituées cette année.

Ce projet fait intervenir de nombreuses compétences techniques dans les équipes : des notions d'IA ou du moins de programmation de comportement pour les robots, de l'analyse d'images et de la communication à l'aide d'un port série R232.

Les robots contrôlés sont des khepera II composés de deux roues motrices, asservies en position ou en vitesse et de nombreux capteurs (proximité, infra-rouge). Dans le cadre de ce projet, le programme est exécuté sur un ordinateur et les commandes sont envoyées au robot par l'intermédiaire d'une liaison série RS232.

2. Règles du jeu

Les principales règles de jeu sont simples:

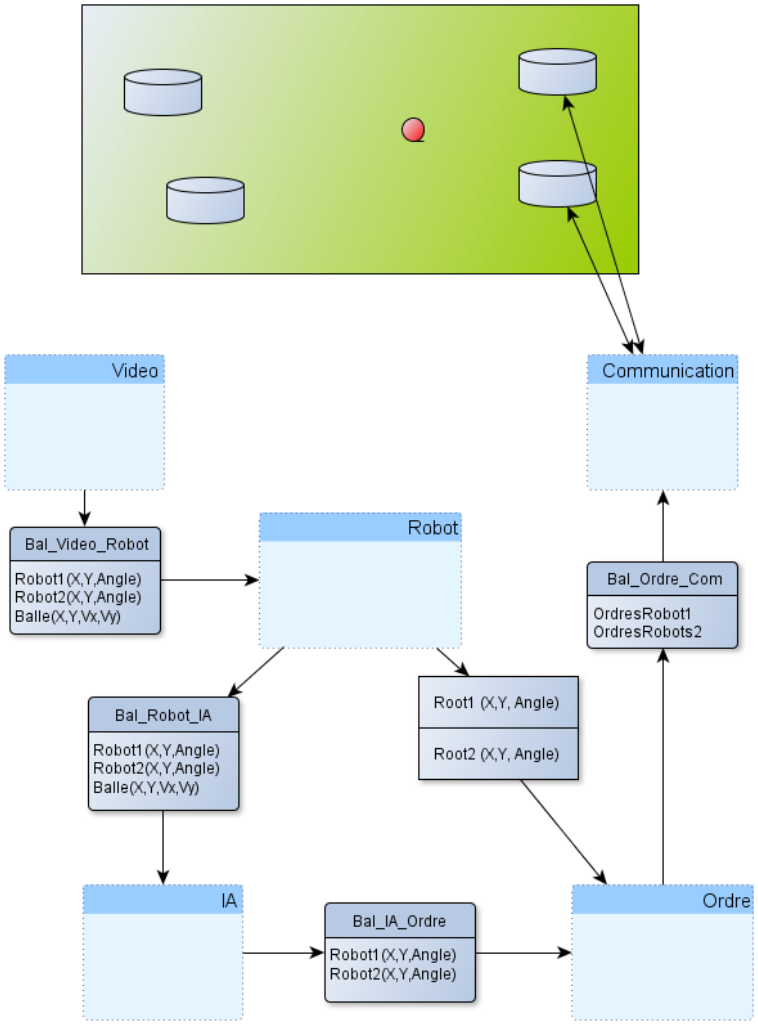
- 2 robots par équipe
- Les robots d'une équipe n'ont pas le droit de dépasser le milieu du terrain
- Un point est marqué lorsque la balle touche le mur du fond du camp adverse.
- Si la balle reste trop longtemps dans un camp. Elle est rendue à l'autre.

Cette année, les robots ont cependant été modifiés pour être dotés d'un "pied" leur permettant de tirer dans la balle en effectuant une rotation. Ce mode de tir est effectivement beaucoup plus efficace que de simplement taper la balle avec le robot comme les années précédentes. Le "pied" du robot a été placé devant afin de rendre les robots symétriques et diminuer le nombre de cas particuliers lors d'un mouvement de tir.

II. Description Technique du projet

1. Organisation générale du code

Chaque années, pour la réalisation de ce projet, les équipes disposent du code des équipes des années précédentes afin de pouvoir aller plus loin et améliorer les défauts de ceux-ci tout en ayant une base sur laquelle s'appuyer. Les codes des années précédentes nous ont cependant semblé étrangement architecturé. Ils étaient découpés en de très nombreuses tâches qui, mal optimisées, entraînaient une très grande consommation CPU (l'un des projets arrivait à consommer jusqu'à 80-90% d'un core i7 installé sur les machines à disposition). Nous avons par conséquent tout refait 'from scratch'. Seule une partie de la communication série avec le robot a été reprise.



Chaque ‘module’ présenté ci-dessus est un thread et les éléments les reliant sont des boîtes aux lettres (Bal_Video_Robot, Bal_Robot_IA...) ou une mémoire partagée (entre Robot et Ordre).
Les structures des messages échangés entre les modules sont donnés de manière simplifiée dans le schéma. Afin de simplifier l’architecture générale, une boîte de message est toujours placée entre deux composants seulement et un seul écrit, un seul lit.

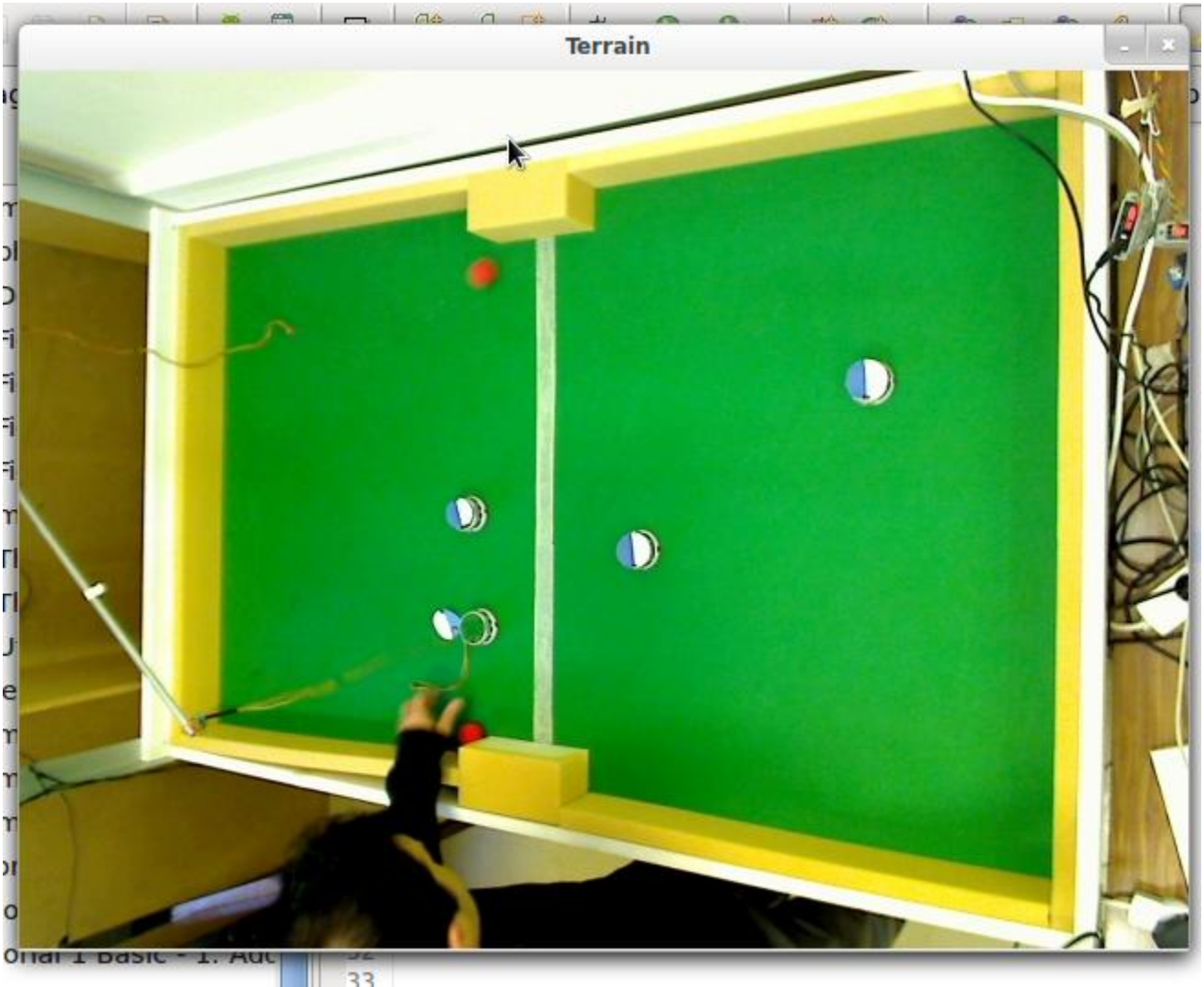
2. Module Vidéo

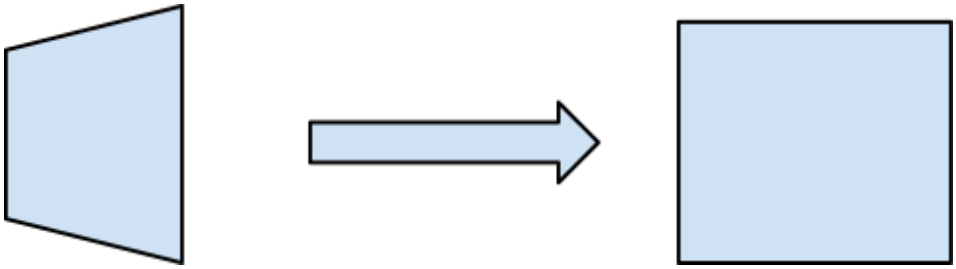
a. Architecture du module

Le module vidéo est implémenté par la classe FieldAnalyse qui gère l’initialisation et la détection du terrain et la transformation des images en la prospective qui nous intéresse pour pouvoir en suite l’analyser pour la détection des objets.

b. Détection du terrain et initialisation du module

Pour ce première étape l’utilisateur spécifie 4 points dans l’image, en suivant l’ordre dans la figure. Ce 4 points sont ensuite utilise pour transformer la prospective et les rapports de l’image pour un traitement optimale.

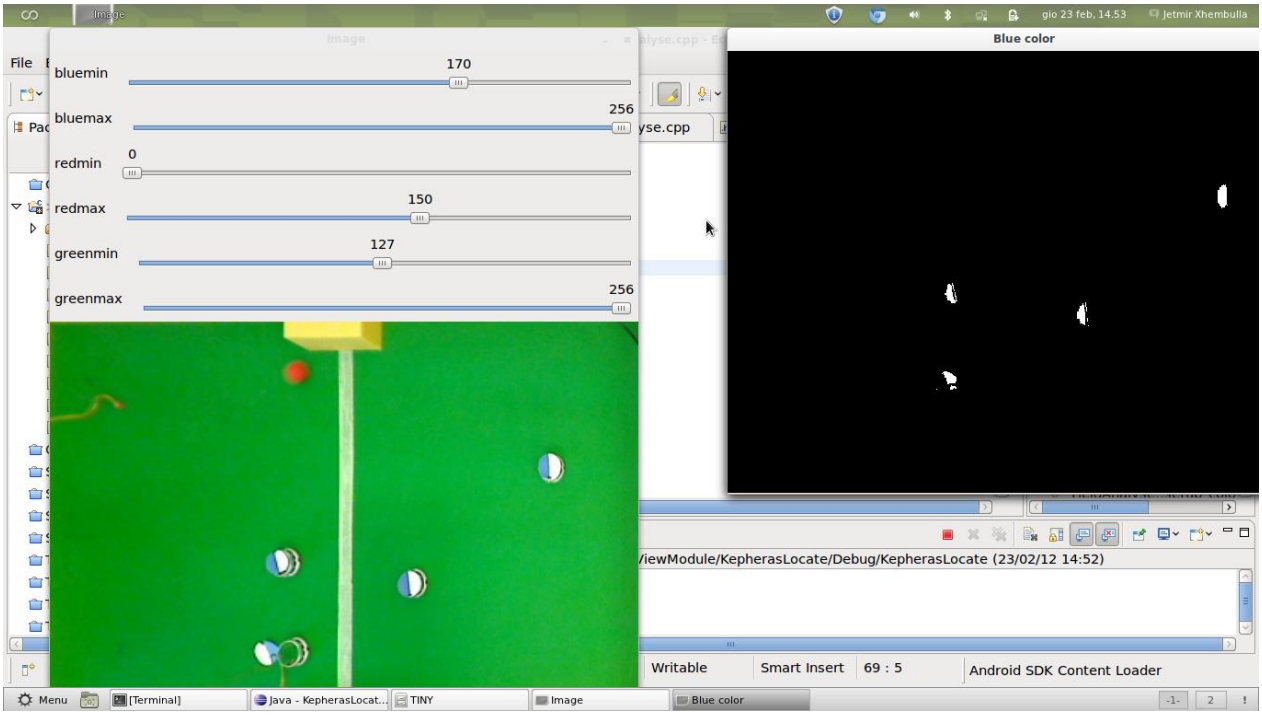




La transformation produit une image rectangulaire du terrain comme si on avait la webcam positionné exactement sur le milieu du terrain. La sélection des 4 points se fait que la première fois et après la positionne des 4 points est sauvegardée dans le fichier « points.txt ». Si on change de webcam ou si on repositionne la webcam il suffit de effacer ce fichier pour sélectionner 4 nouveaux points.

Une optimisation peut être la détection automatique du 4 angles du terrain mais vu que on totalement refait le module vidéo on s'est concentre sur les autres aspects lies a la détection des objets.

Après le lancement de l'application après l'éventuelle détection des points on doit configurer le niveaux des trois canaux RGB à utiliser pour la détection des couleurs rouge, blanche et bleu. Il suffit de changer les valeurs minimale et maximale des trois différentes canaux RGB pour affiner la détection des trois couleurs que nous intéresse.



L'algorithme qui enchaine les différentes détections est le suivant :

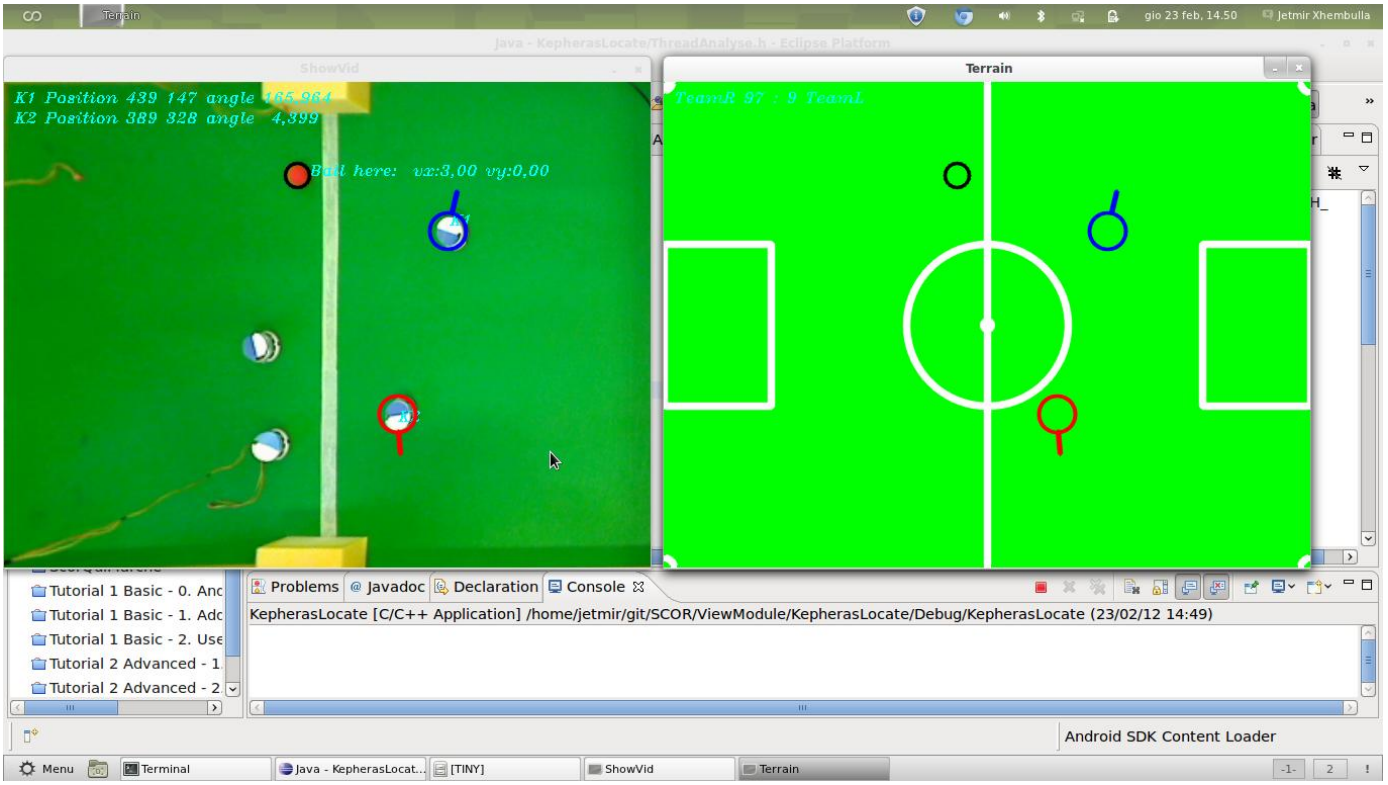
- Préparation pour détecter la couleur bleue et la couleur rouge.
- Déclenchement des trois threads qui analysent l'image en entrée
- Atteinte de la terminaison du travail des deux threads sur les robots
- Préparation pour détecter la couleur blanche.
- Déclenchement des deux threads qui analysent l'image en entrée pour rechercher les robots
- Atteinte de la terminaison du travail des deux threads sur les robots et de celui de la balle
- Envoie du message avec les positions des objets, les angles du kepheras et la vitesse de la balle

On suive ce démarche pour chaque frame envoie par la webcam après avoir transformé l'image en la prospective que nous intéresse.

c. Détection de la balle

La détection de la balle et des deux robots ne s'effectue pas comme les années avant avec le clustering fait par nous-même. Par contre on exploit la librairie OpenCV et on utilise les fonctions qui permet d'avoir les moments d'une image binaire comme l'image suivante.

Le moment de première ordre analyse l'image et nous donne la surface de l'objet présente dans l'image. Avec les autres moments on peut retrouver la position de son barycentre. La détection de la balle est très simple donc. Pour la vitesse on maintient un buffer avec les 10 dernières positions de la balle et pour retrouver sa vitesse on considère la première e la dernière position. Le buffer est mise à jour pour chaque nouveau frame, position.



d. Détection des deux robots

Pour la détection des deux robots la situation est une peux plus complique. On a simplifie la détection en imposant que les robots es trouvent chaque un dans son quart du terrain et que les mêmes ne se déplacent pas dans le partie du terrain qui lui appartient pas. On peut obtenir ça par l'aide du module IA qui oblige les deux kepheras à respecter ce règle nécessaire pour simplifier la détection.
Après avoir détecté les positions des parties blanche et bleue pour chaque robot on calcule le barycentre du robot comme le point au centre du segment qui unit les barycentres du partie blanche et bleue. Pour trouver l'angle de rotation on a pris comme référence (angle 0°) le cas quand la ligne qui sépare les parties blanche et bleue est parallèle au bord supérieure de l'image et la partie bleue est en haut et celle blanche en bas, toujours par rapport au bord supérieure de l'image. Le Kephera en bas par exemple à une rotation de presque 0°. La figure à droite est utile seulement pour avoir une vision plus similaire à un match de foot.

e. Améliorations possibles

Dans les améliorations possibles on peut suggérer de chercher seulement de affiner la détection vidéo ou sinon de implémenter une module vidéo qui soit indépendant du mouvement de la camera pour produire des résultats en sortie. Donc une module vidéo qui s'adapte tout seule aux éventuels mouvements de la webcam. Une adaptation dynamique aux différentes valeurs de lumière peut être aussi intéressante. Sinon le module vidéo est suffisamment complet et ne présente pas de problèmes.

3. Module IA

a. Présentation

Ce module représente le comportement du robot. Dans le cadre de ce projet, il aurait été très intéressant d'un point de vue pédagogique de pouvoir programmer une véritable IA basée sur réseau de neurones et algorithme génétique. Les contraintes de temps ne l'ont cependant pas permis : En effet, pour entraîner cette IA, il aurait fallu disposer d'un simulateur réaliste et pouvoir faire tourner le programme pendant une durée prolongée.

De plus, étant donné la simplicité des règles et des cas à traiter dans un match, un comportement automatique sera probablement plus performant.

b. Fonctionnement

Ce module reçoit du module 'Robot' :

- la position et l'angle de chaque robot de l'équipe
- la position et la vitesse de la balle

Il va, à partir de ces données, déterminer la position à atteindre pour chaque robot.

- Si la balle s'éloigne de notre terrain ou est arrêtée dans le terrain adverse, les deux robots doivent se replacer en défense
- Si la balle est immobile dans notre terrain, le robot situé dans le quart de terrain de la balle doit aller la frapper.
- Si la balle se rapproche de notre terrain, les robots se déplacent sur la ligne de défense afin de l'intercepter.

En défense, nous avons déterminé que l'action la plus efficace était de placer les robots sur une ligne parallèle au fond du terrain et de les déplacer pour intercepter la balle (en gérant la trajectoire et les rebonds sur les murs) toujours sur cette ligne (à la manière d'un pong).

En attaque, le robot fera en sorte de se placer à côté de la balle pour ensuite effectuer une rotation pour frapper. On repère tout d'abord le quart de terrain dans lequel est la balle, puis la position du robot gérant ce quart de terrain par rapport à celle de la balle afin de déterminer si il est préférable de se placer à gauche ou à droite de la balle pour frapper.

Une fois les objectifs des deux robots déterminés, ce module vérifie si ces objectifs ont été modifiés depuis le dernier ordre envoyé et, le cas échéant, envoie un message dans la boîte aux lettres Bal_IA_Ordre. Ce message contient les positions (X,Y,Angle) que chaque robot doit atteindre.

4. Module Ordre

a. Présentation

Ce module est la liaison entre le module "I.A." et le module "Communication". Il est en charge de l'interprétation des objectifs fournis par le module "I.A." Il décompose les "objectifs" fournis par le module "I.A." en ordre élémentaires exécutables par le robot.
Cela permet au module "I.A" de calculer des objectifs sans avoir à se soucier de la réalisation de la trajectoire, et au module "Communication" de donner des ordres aux robots sans avoir à effectuer des calculs superflus.

b. Fonctionnement

Ce module s'exécute en seul thread. Il reçoit des "objectifs" de la part du module "I.A.". Il a été définis 3 objectifs pouvant être communiqué :

- La frappe à gauche : le robot se déplace jusqu'à un point donné, se positionne suivant un angle donné, et effectue une rotation de 270° dans le sens trigonométrique.
- La frappe à droite : le robot se déplace jusqu'à un point donné, se positionne suivant un angle donné, et effectue une rotation de 270° dans le sens horaire.
- Le déplacement : le robot se déplace jusqu'à un point donné, et se positionne suivant un angle donné

Chaque objectif reçu est ensuite transformé en en “déplacement élémentaires”, à savoir un nombre de “crans/pas” pour chaque roue de chaque robot.

Un déplacement est typiquement décomposé de la manière suivante :

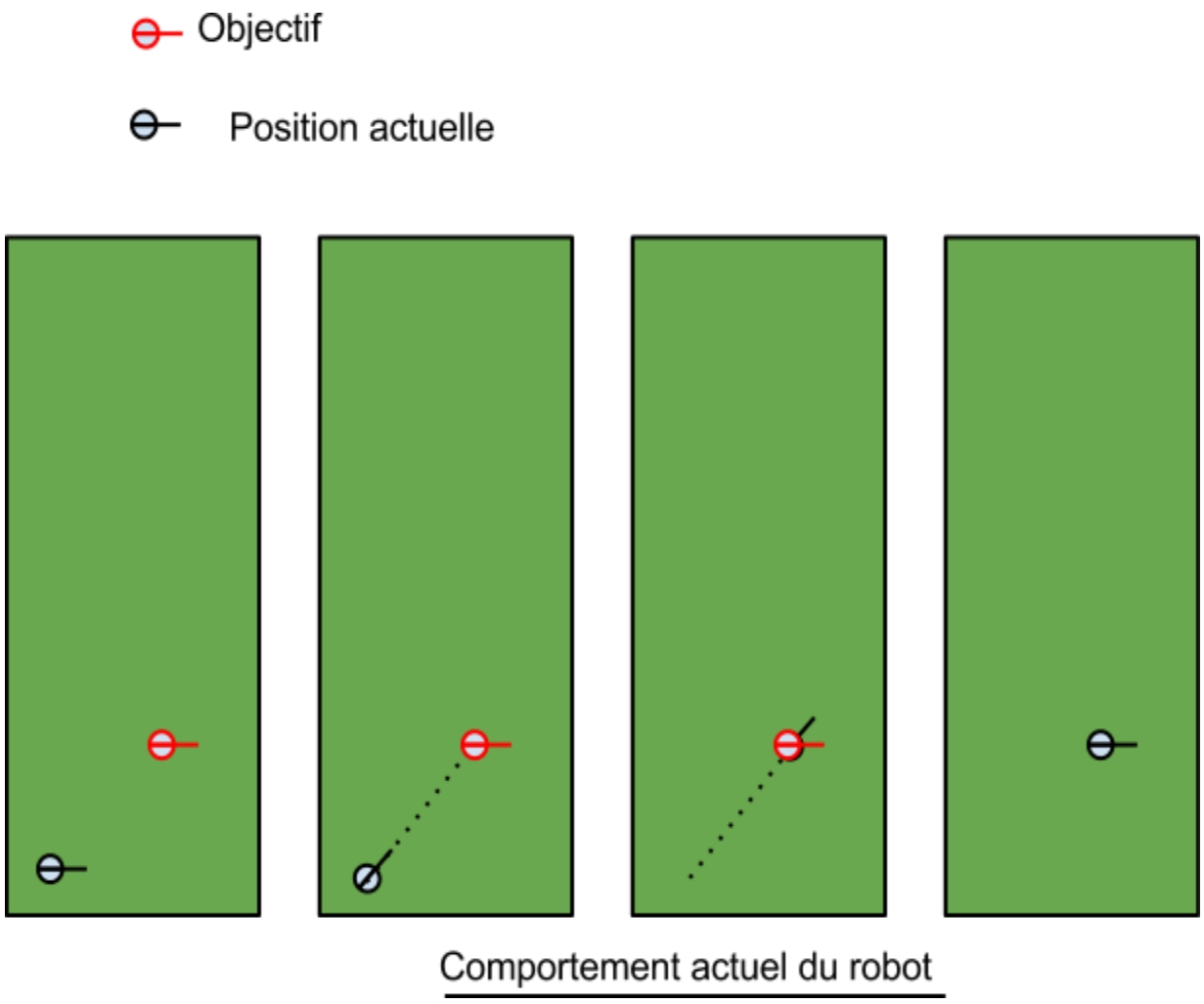
- Une première rotation : destinée à mettre le robot dans la bonne direction pour la suite du mouvement. Elle se présente de la manière suivante :
    - +X pour la roue gauche
    - -X pour la roue droite
    - X pouvant être positif ou négatif suivant le sens de la rotation
- Un déplacement rectiligne : l’objectif de ce déplacement est d’amener le robot au point souhaité. Il se présente de la manière suivante :
    - +X pour la roue gauche
    - +X pour la roue droite
    - X pouvant positif ou négatif, suivant si le robot avance ou recule
- Une seconde rotation : destinée à mettre le robot dans la direction finale donnée. Elle se présente de la manière suivante :
    - +X pour la roue gauche
    - -X pour la roue droite
    - X pouvant être positif ou négatif suivant le sens de la rotation
- Dans le cas d’un tir, une rotation supplémentaire est ajoutée, destinée à effectuer le tir. Elle se présente de la manière suivante :
    - +X pour la roue gauche
    - -X pour la roue droite
    - X pouvant être positif ou négatif, suivant le sens du tir
    - L’angle de rotation lors d’un tir étant toujours le même, la valeur de X est une constante (au signe près) permettant au robot d’effectuer une rotation de 270°.

Ces différents déplacement élémentaires sont placés dans un tableau et transmis par le biais d’une boîte aux lettres au module “Communication”, qui se charge d’effectuer les mouvements.

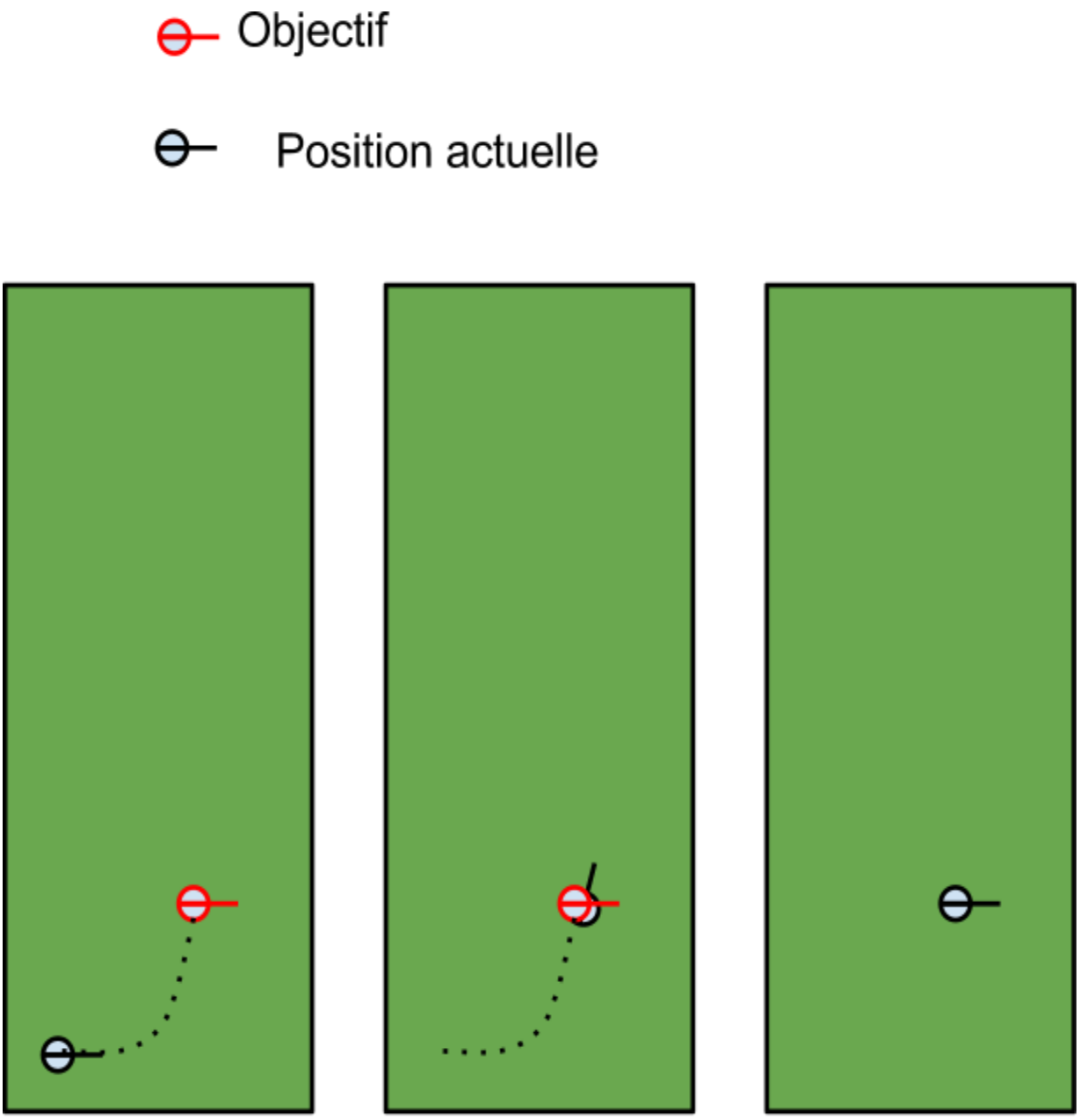
c. Améliorations possibles

Comme dit précédemment, chaque “objectif” est décomposé en “mouvements élémentaires”. Ces mouvements élémentaires nous amènent à décomposer un déplacement en trois “sous-mouvements” (rotation, déplacement rectiligne, rotation).
Une étude plus précise du comportement du robot permettrait peut être de le faire avancer selon des lignes “courbes”, faisant ainsi gagner un temps précieux (la rotation initiale pouvant en effet être grâce à cela supprimée). Ces lignes courbes se présenteraient de la manière suivante :

- +X pour la roue gauche
- +Y pour la roue droite
- X et Y pouvant être positifs ou négatifs suivant le mouvement, bien que probablement de même signes



Comportement actuel du robot



Comportement envisageable du robot

5. Module Communication

a. Présentation

Ce module est chargé, comme son nom l'indique, de gérer la communication série RS-232 entre les robots et l'ordinateur. Il reçoit des commandes du module Ordre et les envoie aux robots.

b. Fonctionnement

Ce module s'exécute en un seul Thread (le temps d'envoi des ordres aux robots peut être considéré comme négligeable, les deux robots sont géré par le même thread). Chaque robot est représenté par son descripteur de fichier dans lequel on écrit les commande et on lit les réponses.

Le module 'Ordre' envoie toujours un tableau de 4 ordres car chaque déplacement du robot d'une position X1,Y1,Angle1 vers une position X2,Y2,Angle2 peut être décomposé en 3 déplacement : une rotation pour cibler l'objectif, un déplacement rectiligne pour l'atteindre puis une rotation pour atteindre l'angle objectif. Le dernier ordre est réservé au cas du tir : il s'agit d'un rotation supplémentaire pour frapper la balle.

c. Améliorations possibles

Actuellement, il faut placer correctement les robots sur le terrain afin que le robot de gauche soir le Robot1 et le robot de droite le Robot2. Il serait utile d'effectuer cette détection dans le code et de pouvoir inverser les robots si les prises USB n'ont pas été branchées dans le bon ordre sur l'ordinateur.

Il arrive que les robots aient des problèmes et se mettent à écrire dans le descripteur de fichier. Actuellement, le code ne prévoit pas ce cas : après l'envoi d'une commande, on lit la ligne suivante du descripteur. Si elle n'est pas correcte, le programme s'arrête. Il faudrait pouvoir vider le buffer en lecture de chaque robot et les réinitialiser dans le code.

6. Module Robot

a. Présentation

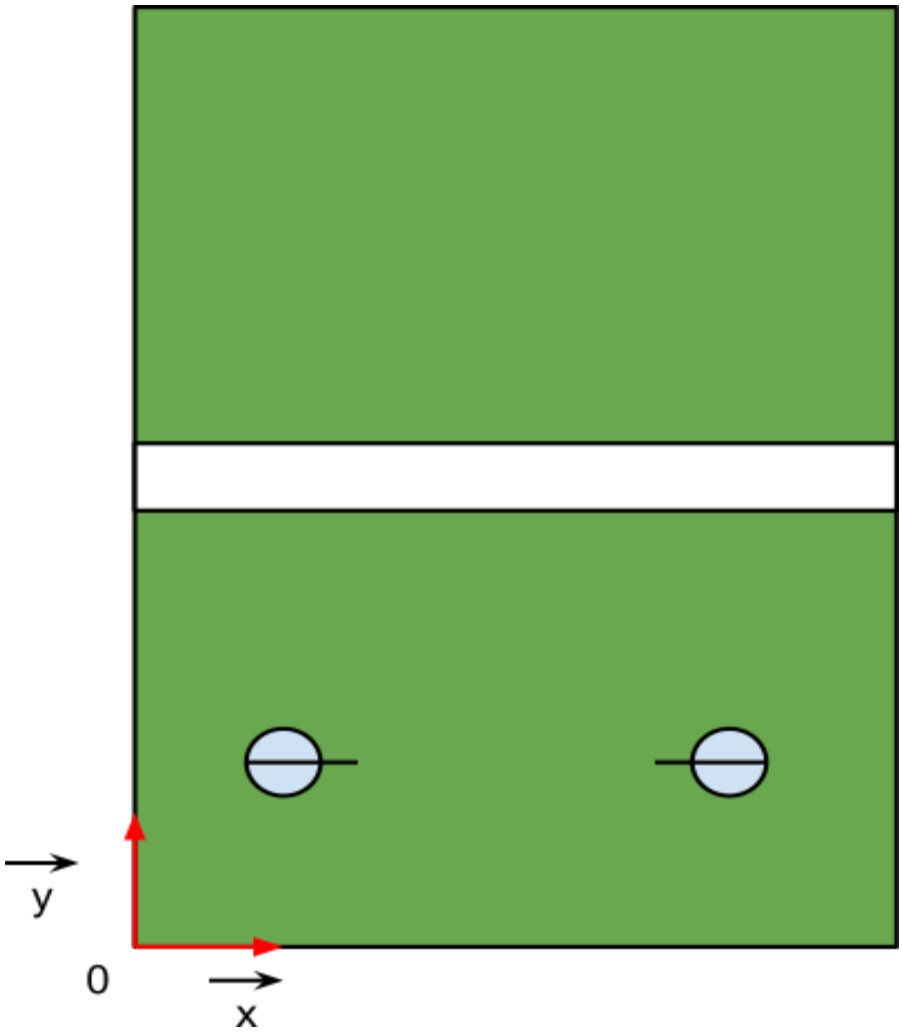
Tout d'abord, le nom de ce module n'est sans doute pas approprié, et n'aide pas à la compréhension.

L'objectif de ce module est de faire une agrégation des informations concernant la position des robots et de la balle, de convertir ces informations fournies dans un repère unique, et de transmettre ces informations au module "I.A.".

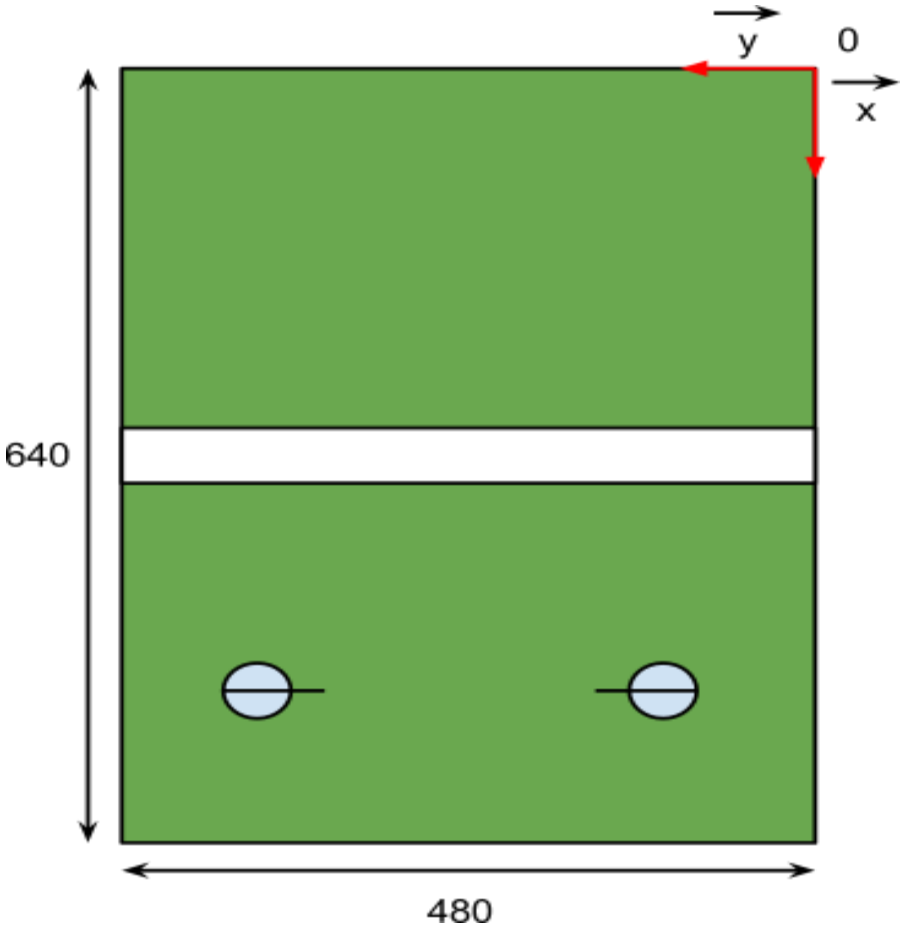
b. Fonctionnement

Ce module s'exécute sur un seul thread.

Le repère utilisé par le module "Robot" (identique à celui utilisé par le module "I.A.") est le suivant :



Le repère utilisé par le module “Vidéo” quant à lui est le suivant :



Lorsque le module reçoit des informations du module “Vidéo”, il est donc nécessaire d’effectuer un changement de coordonnées.

On obtient donc, après changement de repère, les nouvelles coordonnées suivantes :

x = 480 - y_video
y = 640 - x_video
angle = -90 - angle_video
vitesse_x = - vitesse_y_video
vitesse_y = - vitesse_x_video

Ces nouvelles données sont stockées dans une mémoire partagée utilisée par le module “Ordre”.

Initialement, lorsque le module recevait des informations du module “Communication”, le module mettait à jour les informations reçues dans la mémoire partagée. Une conversion (empirique) était effectuée afin de transformer les “pas de robots” transmis par le robot en déplacement de robot dans le repère utilisé par le module “Robot”.
Ces “déplacements de robot” sont des déplacements selon l’axe des x, l’axe des y, et une variation d’angle dans le repère du module.

Cependant, les informations transmises par les robots étant la majorité du temps erronées, cette idée a été abandonnée, le calcul n'est donc pas détaillée ici.

Lorsque les coordonnées d'un robot, ou de la balle sont mise à jour, l'ensemble des coordonnées des robots et de la balle (ainsi que la vitesse pour cette dernière) sont transmises au module "I.A." par le biais de la boîte au lettre correspondante.

c. Améliorations possibles

La possibilité mettre à jour la position des robots grâce aux informations transmises par les robots permettrait éventuellement d'allouer moins de ressources au module "Vidéo", et permettrait également d'obtenir une meilleure estimation en temps réel de la position de chacun des robots.
Il serait donc judicieux de déterminer pourquoi le robot ne renvoie des informations correctes concernant son déplacement, et éventuellement de corriger cela, afin de permettre une meilleure fréquence de mise à jour de la position des robots.

III. Conclusion

Pour conclure, nous avons trouvé ce projet très intéressant. Tout d'abord, l'aspect ludique du sujet est très attirant : l'objectif étant de programmer des robots pour effectuer un match.

La liberté dans les méthodes et outils utilisés ainsi que dans la gestion de projet est également très appréciable. Ce projet nous laisse totalement autonome avec pour seul objectif de parvenir à faire jouer un match à nos robots lors de la présentation.

Une bonne organisation au sein de l'équipe est essentielle et c'est dans cette partie que réside une grande partie de la difficulté du projet. Nous avons eu également de nombreux problèmes à cause de certaines erreurs générées par les robots qui semblent se déclencher plus ou moins aléatoirement et faisaient en général planter le programme avant de pouvoir effectuer des tests complets.

Au final, le projet SCOR nous a beaucoup plu pour les raisons citées plus haut. Il est très différent des projets de conception proposés en 5IF et nous offre une toute autre approche du travail en équipe et de la conduite de projets.