Projet robotique autonome :

MLP Ball catcher

Victor Cheimanoff, Thibaud Cheippe, Yoan Mollard

13 février 2013

Table des matières

1 Introduction

L'objet de ce projet est d'utiliser un bras robotique pour rattraper une balle qui lui serait lancée par un utilisateur. Le système utilise un équipement de motion tracking permettant de repérer la balle dans l'espace à l'aide de caméras. Une fois la trajectoire de la balle amorcée il calcule un point d'impact puis ordonne au bras de s'y positionner.

Bien que la main robotique soit capable de se fermer il semble important d'omettre ce détail dans un premier temps pour se consacrer sur l'obtention d'un impact entre la balle et le poignet. Ceci nottamment car orienter le bras dans une configuration où la balle peut rentrer au milieu des doigts ajout une difficulté supplémentaire.

Afin de conclure cette introduction nous présentons rapidement le matériel utilisé, à savoir le bras robotique et le système de capture Optitrack. L'ensemble de la scène est représenté en figure 2.

1.1 Le bras robotique

Le bras utilisé est un bras anthropomorphe à taille réelle conçu par Rhoban Project. Il dispose de sept servomoteurs Dynamixel immitant le bras humain. Il vient avec une suite logicielle conçue par l'équipe de Rhoban Project permettant de le manipuler, qui va consiter en trois éléments :

- Une interface graphique, RobotBoard, qui permet de créer et d'enregistrer des mouvements, que nous n'utiliserons pas dans ce projet.
- Le RhobanServer qui gère la communication bas niveau avec le matériel
- Le SDK Rhoban, qui nous permet d'envoyer des instructions au robot, ainsi que de lire facilement les valeurs prises par les moteurs du robot, qui sera l'outil principalement utilisé pour notre projet.

1.2 Le système de capture « Motion tracking »

Le système de capture utilisé est composé de trois caméras Optitrack VX1000. Equipées d'un éclairage infrarouge ces caméras, lorsqu'elles sont convenablement disposées autour d'une scène, permettent de localiser dans l'espace des boules réfléchissantes d'un centimètre de diamètre. Seuls les marqueurs réfléchissants fournis sont censés pouvoir être tracké par Optitrack. Cependant nous avons pensé à recouvrir notre balle de papier d'aluminium afin de substituer la balle aux marqueurs.

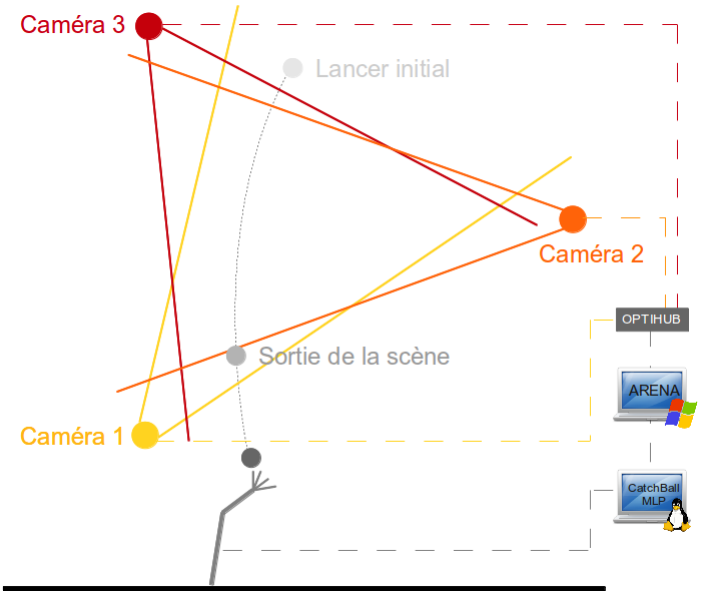

FIGURE 1 – Schéma de la scène : bras accompagné des trois caméras et leur champs de vision, balle et sa trajectoire, le serveur Windows sur lequel foncitonne Arena et auquel sont connectées les caméras et le client Linux composé de trois modules et pilotant directement le bras

2 Découpage du projet

Nous avons décomposé le projet en trois parties distinctes :
- La mise en place du système de capture Optitrack de manière à obtenir les coordonnées x, y, z d'un point de l'espace
- L'obtention du modèle géométrique inverse du robot, le calcul de la trajectoire de la balle et la planification du point d'impact
- La représentation 3D (OpenGL) de la scène pour visualiser la trajectoire calculée, le point d'impact choisi et éventuellement la configuration du

robot

2.1 Trois exécutables distincts

L'idée est de conserver ces trois parties indépendantes dans trois exécutables différents, puis de les faire communiquer sur leurs entrées et sorties standard en utilisant un protocole texte très simple. Les trois exécutables sont :

- Le client récupérant les coordonnées diffusées par Arena (Client/Data Streaming)
- Le calculateur de la trajectoire de la balle planifiant le point d'impact (Impact)
- L'affichage en temps réel (Viewer)

Ce découpage est représenté sur la figure 2. On lance ainsi le programme final en redirigeant les sorties au bon exécutable :

```
~$ ./Client | ./Impact | ./Viewer
```

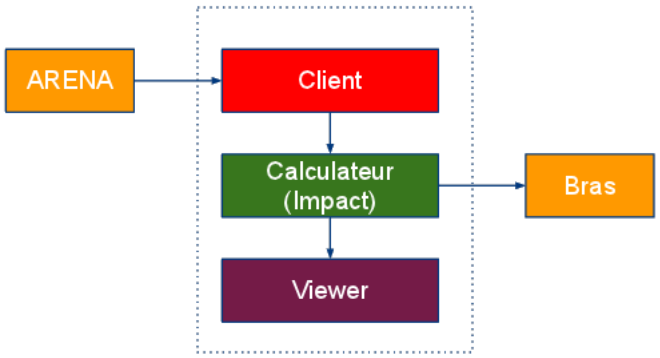

FIGURE 2 – Découpage de l'application en trois modules : données Arena en entrée et commande du robot en sortie

2.2 Le protocole

Il est nécessaire de définir un protocole basique de communication entre les trois exécutables. Bien qu'il n'ait pas encore été spécifié, voici un exemple de ce qui pourrait être utilisé :

- Entre le client et le calculateur, on transmet pour chaque instant t les coordonnées x, y, z de la balle dans le repère de la scène sur une nouvelle ligne :

```
t       x       y       z
```
Par exemple :
```
0.01    45.3    24.8    176.4
0.02    45.9    23.9    174.2
0.03    47.5    24.1    171.7
0.04    51.0    23.2    167.9
...     ...     ...     ...
```
–

– Entre le calculateur et le visualisateur, on inclue les précédents relevés t, x, y, z de position de la balle ainsi que les paramètres de trajectoires A_x, A_y, A_z, B_x, B_y, B_z. On identifie une trajectoire grâce à un entier "id" pour pouvoir aisément la mettre à jour lors de l'obtention d'un nouveau point :
```
t       x       y       z
id      ax      ay      az      bx      by      bz      // Mise à jour ou création
                                                        // de la trajectoire numéro "id"
```
Par exemple :
```
0.01    45.3    24.8    176.4
0.02    45.9    23.9    174.2
0.03    47.5    24.1    171.7
0.04    51.0    23.2    167.9
1       18.6    34.8    44.9    33.12   178.6   8.7     // Création de la trajectoire 1
0.05    42."    24.9    170.3
0.06    51.7    23.4    167.2
1       17.6    34.8    44.9    33.12   178.6   8.7     // Mise à jour de la trajectoire 1
0.07    49.9    23.9    176.2
```

3 Exportation des coordonnées d'un point avec Optitrack

Cette partie est un tutoriel de mise en route rapide du système de capture, principalement à destination des futurs utilisateurs ou repreneurs du projet. Le fabriquant du système documente peu ses produits mais les tutoriels sont plutôt utiles [1].

La première étape consiste à disposer les caméras autour de la scène. Six caméras sont fournies dans le pack, mais nous avons commencé par en monter seulement trois sur trépied. Selon la qualité des résultats obtenus il est possible d'ajouter des trois autres à mi-hauteur sur les trépieds pour augmenter la précision des coordonnées obtenus.

Les trois (ou les six) caméras sont connectées à un hub USB (Optihub) lui-même connecté à un PC sur lequel est installé le logiciel Arena. Il est important de disposer les caméras plutôt vers le début de la trajectoire de la balle, car c'est

ici qu'il nous reste suffisamment de temps pour effectuer les calculs et surtout positionner le bras robot à la position calculée.

3.1 Calibration des caméras

La première étape consiste à calibrer les caméras entre elles. En démarrant l'utilitaire de calibration d'Arena on commence par jouer sur les paramètres ainsi que sur l'orientation des caméras. Selon nos essais, il est préférable d'éviter que les caméras se voient les-unes-les-autres et de choisir une intensité faible pour les LEDs infrarouge : une trop forte intensité augmente le nombre de parasites. Avant de continuer il faut s'assurer qu'aucun point parasite n'est détecté par l'une des caméras en l'abscence de marqueur dans la scène. Les parasites peuvent être des objets réfléchissants ou brillants (fenêtre, lumière, métal ...). Afin de comprendre d'où proviennent les parasites il est utile d'afficher les images noir et blanc renvoyées par les caméras (clic droit sur une frame puis Grayscale image). Avant de commencer la calibration aucun parasite ne doit apparaître en l'absence de marqueur et un marqueur placé dans la scène doit être reconnu par les trois caméras.

L'étape de calibration peut ensuite commencer. Il s'agit de déplacer trois marqueurs au bout d'une baguette Optiwand pour couvrir au maximum le volume de la scène dans un maximum d'orientations possibles. Toutes les caméras doivent visualiser les marqueurs pour que les points puissent participer à la calibration. A l'issue du processus de calcul Arena est capable de disposer les caméras dans l'espace les unes par rapport aux autres sur une carte.

Il faut ensuite placer le repère (représenté par une équerre en métal sur laquelle sont disposés trois marqueurs) afin de définir l'origine et l'orientation des axes du repère. La calibration est ainsi terminée et Arena peut afficher la position d'un marqueur dans l'espace.

Note : la licence Arena a été enregistrée pour la caméra 179282. Cette caméra doit donc impérativement être branchée pour démarrer Arena.

3.2 Constitution d'un Corps Rigide

Une fois les caméra calibrées, il est important de créer un corps rigide pusiqu'il s'agit a priori du seul moyen d'exporter les coordonnées des points capturés.

Un corps rigide représente un ensemble de points ne bougeant pas les uns par rapport aux autres. Cela nous permet de reconnaître facilement les points intéressants (comme les trois marqueurs de la baguette de calibration). De plus, il est impératif d'avoir un corps rigide afin de pouvoir transférer les positions à un programme tiers. La création d'un corps rigide se fait facilement en faisant bouger les marqueurs de l'objet dans le champs de vision des caméras.

3.3 Exportation des coordonnées

Arena embarque un serveur diffusant les coordonnées du corps rigide, apellé NatNet. Un client récupérant ces coordonnées existe déjà pour Windows, nous l'avons adapté pour qu'il puisse fonctionner sous Linux et pour l'intégrer à notre Ball catcher. Il correspond à la partie "Client" dans notre découpage.

3.4 Calcul de l'erreur sur les coordonnées

Afin de connaître la précision du système de motion tracking, nous avons mesuré l'écart entre deux marqueurs avec une règle, puis avec les coordonnées récupérées depuis le logiciel Arena. En moyenne, on obtient une erreur de 3 millimètres tous les 25 centimètres

4 Modélisation et Modèle Géométrique Inverse

Nous désirons déplacer le robot en lui fournissant un position de destination absolue (x, y, z) dans un repère lié à la salle. Or, le robot se positionne en utilisant les valeurs angulaires que doivent prendre ses moteurs. Il était donc nécessaire d'implémenter un module permettant de passer d'une position absolue aux valeurs angulaires associées, d'où la nécessité du calcul d'un modèle géométrique inverse. Ceci est intégré dans la partie "Calcul d'Impact" dans le découpage de notre application.

4.1 Modélisation

Afin de modéliser le bras, nous avons nommé les moteurs comme indiqué sur la photographie 3 :

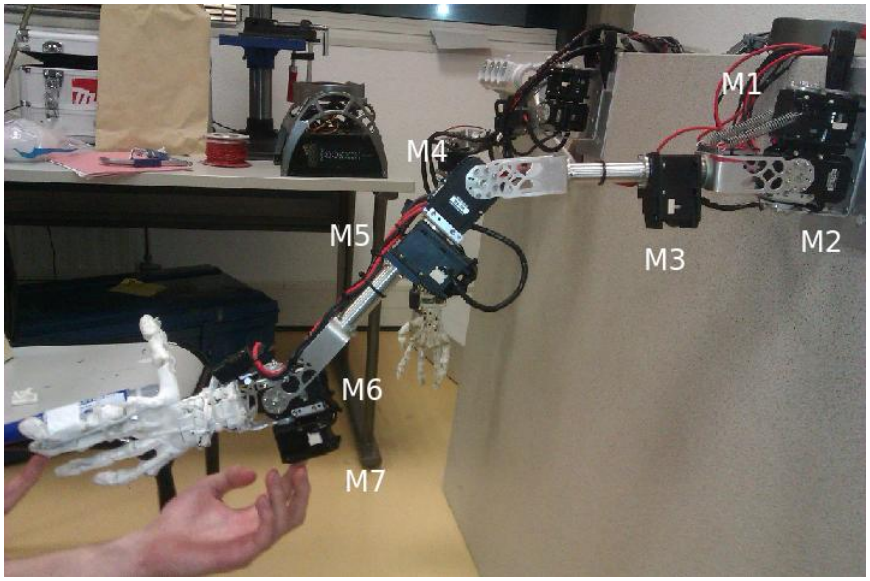

FIGURE 3 – Bras robotique et noms des moteurs

La première étape de la modélisation a été de mettre en place une paramétrisation de Denavit-Hartenberg, qui nous permet de déterminer un repère différent lié à chacun des moteurs, afin de pouvoir récupérer ensuite des équations exploitables pour la détermination du modèle géométrique inverse.

La représentation en question est en figure 4 :

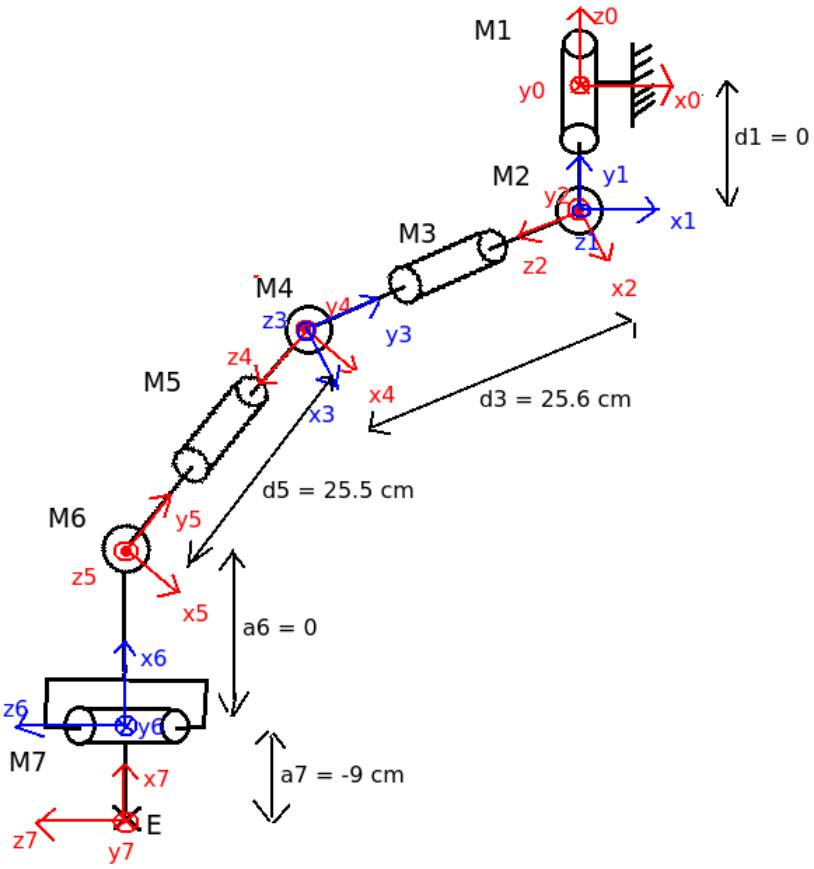

FIGURE 4 – Représentation du bras en paramétrisation de Denavit-Hartenberg

Sur cette représentation, le point E représente la position de l'effecteur, au centre de la main robotique, point que nous voulons être celui à réceptionner la balle. Nous avons arbitrairement décidé de la valeur de sa distance au moteur 7.

Le repère R0 représente le repère lié à la salle. Pour des raisons de clareté, il a été représenté au centre du moteur 0, néanmoins, dans la paramétrisation de denavit Hartenberg, seul l'orientation de son axe z est imposé. Nous avons donc fais le choix de placer son origine au centre du moteur M2, ce qui nous permettra de simplifier nos équations en forçant ainsi $d1 = 0$.

Une fois cette représentation effectuée, et ces repères placés, nous sommes en mesure d'établir le tableau de Denavit-Hartenberg suivant :

i, i+1	Rot_{zi}	$Trans_{xi+1}$	$Trans_{zi}$	Rot_{xi+1}
0, 1	θ_1^*	/	d1 = 0	$+90^o$
1, 2	θ_2^*	/	/	$+90^o$
2, 3	θ_3^*	/	d3 = 26.5 cm	-90^o
3, 4	θ_4^*	/	/	$+90^o$
4, 5	θ_5^*	/	d5 = 25.5 cm	-90^o
5, 6	θ_6^*	a6 = 0	/	$+90^o$
6, 7	θ_7^*	a7 = -9 cm	/	/

Une fois cette représentation terminée, nous pouvons commencer l'implémentation de la détermination d'un modèle géométique inverse.

4.2 Détermination des équations du modèle géométrique inverse

Afin de déterminer un modèle géométrique inverse, la première étape a été de trouver et de définir les équations à résoudre permettant d'obtenir les valeurs angulaires à prendre sur les moteurs pour correspondre à une position et une orientation fixée de la main.

Pour ce faire, nous avons utilisé le logiciel libre de calcul formel Maxima, dans lequel nous avons défini les matrices de transformations homogènes associées au bras grâce à notre représentation de Denavit-Hartenberg, ainsi qu'une matrice de positionnement et d'orientation désirée de la main dans le repère 0, qui correspond à la position et à l'orientation de la main que l'on souhaite obtenir, et qui s'écrira avec le triplet (x, y, z) qu'un modèle d'interpolation avec le module de prédiction de trajectoire fournira.

Constatons que pour une orientation et une position fixée de la main, bien qu'il y ait une infinité de positions du bras qui soient compatibles, la valeur de l'angle du moteur 4 reste elle toujours identique. De plus, nous savons que la distance entre le centre du moteur 2 et le centre du moteur 4 est fixe, dépendante des caractéristique du robot, égale à d3. Il en va de même pour la distance entre les centres des moteurs 4 et 6, qui vaut d5.

De plus, pour une position et une orientation fixe de la main, le centre du moteur 6 est lui aussi déterminé, et immobile dans le repère 0. Le centre du moteur 2 étant l'origine de ce repère 0, la position désirée de l'effecteur nous donne connue la position du centre de ce moteur 6, et donc la distance entre les centres des moteurs 2 et 6.

Nous avons donc ainsi un triangle formé par les centres des moteurs 2, 4 et 6, dont nous connaissons les longueurs et côtés, ce qui nous permet ainsi de déterminer la valeur de l'angle du moteur 4 selon les paramètres imposés par la position désirée de la main.

Arrivé à ce niveau la, nous avons donc 6 valeurs à déterminer, pour lesquelles nous avons 1 degré le liberté. Nous faisons le choix arbitraire d'imposer la valeur du moteur 3 (choisi du fait que le moteur 5 permet une liberté suffisante dans le déplacement malgré une immobilisation du moteur 3), ce qui nous fige le système.

Après quoi, nous utilisons maxima pour obtenir des équations sur les angles des moteurs restants. Ainsi, nous pouvons déterminer (toujours en fonction des coordonnées désirées de la main) des équations nous permettant d'avoir la valeurs des angles restant en recherchant et en exprimant les égalités suivantes avec Maxima :

- θ_4 est connu, déterminé précédemment via la méthode du triangle.
- θ_3 est fixée arbitrairement.
- L'expression de l'égalité $O_{2,6} = T_{2,0}.Od_{0,6}$ nous donne deux équations de Paul de type 2 nous permettant de déterminer respectivement θ_1 et θ_2
- L'expression de l'égalité $T_{4,7} = Td_{4,7}$ nous permet enfin de récupérer trois équations qui nous donneront les valeurs de θ_5, θ_6 et θ_7

Nous avons donc, à ce niveau, fixé arbitrairement la valeur de M3, et obtenu six équations nous permettant de déterminer les valeurs des moteurs restants.

4.3 Résolution des équations du modèle géométrique inverse

Nous avons ensuite entamé l'implémentation d'un module du code C++ qui nous permettra de résoudre ces équations, afin d'obtenir les valeurs angulaires à passer effectivement au robot afin que la main se place dans une position (x, y, z) voulue.

Les équations à résoudre par ce module sont toutes des équations de Paul de type 2 ou des équations triginométriques, qui auront toutes, dans le cas ou l'équation est bien posée, et dans le cas général, deux solutions distinctes, toutes deux mathématiquement correctes, mais dont seule une sera applicable sur le robot. Ainsi, il est necéssaire pour ce module de tester les valeurs renvoyées par les fonctions de résolution d'équations afin de savoir quelle valeure garder, et pour se faire, de récupérer toutes les valeurs minimales et maximales pouvant être prises par les différents moteurs composant le bras.

Faute de temps, l'implémentation de ce module n'a pas pu être terminée, néanmoins, nous disposons de tous les éléments nécessaires à son implémentation,

et n'est donc pas fonctionnelle uniquement par manque de temps.

5 Prédiction de trajectoire

Nous avons implémenté un module C++ permettant de calculer et de prédire la trajectoire d'un projectile à partir des données relevées par le système Optitrack. Ce module suivant est également intégré dans la partie "Calcul d'Impact", il permet de calculer et de prédire la trajectoire d'un projectile à partir des données relevées par le système Optitrack.

5.1 Modèle physique

Nous nous sommes placé dans le cadre de la mécanique Newtonienne, en considérant notre projectile comme un objet ponctuel, étant uniquement soumis à la force de gravitation terrestre, et en négligeant donc les forces frottements.

En considérant un repère orthonormé lié à la salle, dont l'axe z serait vertical, orienté vers le haut, nous avons donc ainsi des équations de trajectoire de la forme suivante :

$x = A_x t + B_x$

$y = A_y t + B_y$

$z = -\frac{1}{2} g t^2 + A_z t + B_z$

Ou les A_i et B_i sont des constantes d'intégration qui apparaissent lors du calcul de ces équations, et que nous cherchons donc à déterminer afin de prédire cette trajectoire.

Nous avons donc à ce niveau un système de trois équations à six inconnues, il nous faudra donc, d'un point de vue mathématique, et donc sans tenir compte des imprécisions du module de capture, les valeurs de positions du projectile à deux instants différents (ie : deux quadruplets (x_i, y_i, z_i, t_i)) afin de calculer les valeurs de ces constantes, et donc d'avoir l'équation de trajectoire du projectile.

Avec deux tels quadruplets, nous obtenons les valeurs suivantes :

$A_x = \frac{x_j - x_i}{t_j - t_i}$

$A_y = \frac{y_j - y_i}{t_j - t_i}$

$A_z = \frac{(z_j + \frac{1}{2} g t_j^2) - (z_i + \frac{1}{2} g t_i^2)}{t_j - t_i}$

$B_x = x_i - A_x t_i$

$B_y = y_i - A_y t - i$

$B_z = z_i + \frac{1}{2} g t_i^2 - A_z t_i$

5.2 Implémentation pratique

Néanmoins, dans la pratique, les valeurs relevées par le module de motion tracking ne sont pas des valeurs éxactes. Une imprecision dans la mesure, même faible, peut avoir des répercutions importantes sur les points éloignées d'une trajectoire qui serait modélisée à partir de ces seules mesures.

Afin de palier à ce problème, le module de prédiction de trajectoire va effectuer plusieurs calculs des solutions des équations de mouvements, à partir de différents quadruplets de points envoyés par le système de motion tracking, et utiliser pour sa prédiction une moyenne arithmetique non pondérée sur les différentes valeurs ainsi calculées, ce qui nous permet de réduire l'erreur en augmentant le nombre de mesures.

Ainsi, en effectuant un nombre de mesure suffisant avec le système de motion tracking, nous devrions être en mesure d'obtenir une prédiction de trajectoire réaliste.

6 Représentation 3D de la scène, du bras, de la balle et de sa trajectoire

Nous avons ajouté au projet un dernier module "visualisateur 3D" dont le rôle est de représenter la scène en 3D en temps réelle. Le minimum, et de loin le plus important est de représenter le système de coordonnées de la scène (défini par le Calibration Square), la trajectoire prévue et affinée à chaque instant, et le point d'impact calculé.

Le visualisateur utilise les bibilothèques OpenGL et GLUT pour QT. Les trois fichiers *main*, *window* et *glwidget* s'occupent du fonctionnement global de l'application, de la fenêtre, et du composant OpenGL. Ils sont issus d'un exemple de QT pour OpenGL sous licence BSD. La scène est dessinée dans $GLWidget :: paintGL()$, méthode appelée à chaque rafraichissement de l'image. Les éléments de la scène (balle, sol, mur, robot ...) sont déclarés dans l'en-tête de GLWidget et dessinés dans cette méthode $paintGL()$. Les éléments affichables sont :

- *CoordinateSystem* : Cet élément représente le système de coordonnées de la scène. Il est déplacé en fonction du système de coordonnées OpenGL pour correspondre à celui réellement utilisé dans la scène.
- *Ball* : Cet élément représente une balle dont la couleur, le diamètre et la position dans l'espace sont moifiables
- *Plane* : Cet élément représente un plan, typiquement utilisé pour le sol et les murs. Il peut être dessiné soit avec un fond plein, soit avec un fond cadrillé et transparant
- *Arm* : Cet élément représente le bras, la position de la base et les angles des servomoteurs peuvent être modifiés. Le dessin fait appel au modèle géométrique.

Il est possible de déclarer autant d'exemplaires de bras, de balles ou de trajectoires que souhaité. Les trajectoires disposent d'un identifiant pour facilier leur mise à jour au cours du temps.

La figure 5 est une capture du visualisateur, sur une scène comprenant un sol cadrillé, un mur gris plein, le repère scène à son pied à gauche (axe x rouge, axe y vert, axe z bleu), ainsi qu'une trajectoire et une balle. Sur cette capture le bras n'est pas encore modélisé.

7 Conclusion

Nous avons découpé ce projet en trois parties : l'obtention des coordonnées avec Arena, réalisée par le module "Client/Data streaming", le calcul de la trajectoire de la balle et du point d'impact réalisé par le module "Impact" qui commande le bras en faisant appel aux modèles géométriques direct et inverse

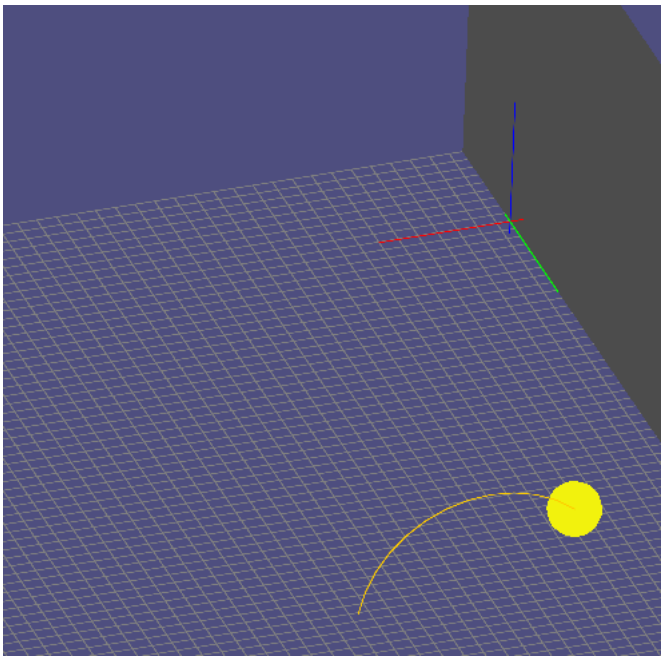

FIGURE 5 – Capture de la visualisation OpenGL d'un repère, un sol, un mur, une trajectoire et une balle

du robot, et un visualisateur 3D permettant de représenter la scène.

Les trois parties ne sont pas encore terminées et demandent a être encore approfondies avant d'être rassemblées et qu'elle puissent communiquer ensemble grâce au protocole simple décrit dans ce rapport. L'utilisation obligatoire des marqueurs pour l'aquisition des points avec Optitrack nous empêche d'utiliser directement une balle. Même recouverte de papier aluminium les résultats ne sont pas satisfaisant, car plutôt que de détecter un point l'aluminium reflète plusieurs points qu'il est impossible de regrouper dans un corps rigide étant donné qu'il n'y en a pas toujours le même nombre. Une parade reste donc encore à trouver pour utiliser Optitrack avec une balle.

Faute de temps, l'implémentation du module de modélisation n'a pas pu être terminée. Néanmoins, nous disposons de tous les éléments nécessaires à son implémentation. Le visualisateur est pour l'instant capable d'afficher les principaux éléments graphiques, mais il ne s'agit que d'affichage. La source des données du visualisateur sont les autres modules : elles proviennent des informations transmises par Optitrack ou relayées/calculées par le calculateur de point d'impact. D'autre part la classe permettant de représenter le bras est pour le moment très incomplète, il lui faut s'appuyer sur le modèle géométrique

direct pour dessiner correctement les seux segments représentant graphiquement le bras dans le modèle 3D.

Références

[1] Tutoriels NaturalPoint Optitrack
`http://www.naturalpoint.com/optitrack/products/arena/tutorials.html`