UNIVERSITÉ DE LORRAINE | IUT nancy Charlemagne Informatique

Robot billard joueur de football

Antoine FONTANEZ - Corentin FROGER - Nathan PIERROT

Tuteur : Pierre-André GUENEGO

Table des matières

Robot billard joueur de football **1**
Réécriture détaillée du sujet **3**
1. Robot Physique 3
2. Environnement de Jeu 4
3. Interface Utilisateur 4
4. Serveur 4
5. Simulateur 4
Étude de l'existant **5**
1. Teutonic 5
2. PR2 de Willow Garage 5
3. Deep Green 5
Étude technique **7**
1. Robot 7
a) Langage de programmation 7
b) Librairies 7
2. Interface web 7
a) Langages de programmation 8
b) Librairies 8
3. Serveur 8
a) Langage de programmation 8
b) Librairies 9
4. Simulateur 9
a) Langage de programmation 9
b) Librairie 9
Problèmes pouvant être rencontrés : **10**

Réécriture détaillée du sujet

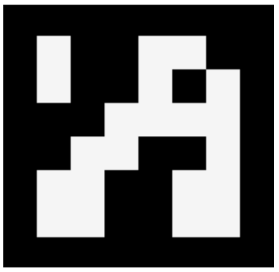
Le projet consiste en le développement d'un système de robot autonome pour jouer au billard. Les objectifs de ce projet sont de connecter le robot à un serveur que nous développerons afin qu'il puisse recevoir des ordres de celui-ci. Une interface web sera développée pour visualiser l'état des robots et les commander en temps réel. En complément, des simulateurs seront créés pour reproduire des environnements de billard et de robotique en essaim, facilitant ainsi les tests et la validation des fonctionnalités du système.

Ce robot évolue sur un billard équipé d'une caméra prenant les informations visuelles et est contrôlé par une interface utilisateur sur navigateur web. Voici les éléments essentiels :

1. Robot Physique

Le robot est un modèle simple roulant, construit sans capteur additionnel ni actionneurs autres que des roues motorisées. Il n'a donc ni bras ni éléments humanoïdes.

Pour sa détection et son suivi, le robot est chapeauté par un marqueur ArUco, qui est une sorte de QR-Code de reconnaissance visuelle couramment utilisé en robotique. Ce marqueur permet à la caméra de localiser le robot avec précision et d'orienter ses actions en fonction de sa position.



Exemple d'ArUco

2. Environnement de Jeu

Le robot évolue sur une table de billard qui sert de terrain de jeu, avec des boules de billard disposées aléatoirement sur sa surface, permettant d'imaginer une interaction simulant une partie de football.
Une caméra fixée au plafond capture l'image de la table de billard. Cette caméra est essentielle pour suivre en continu le robot et détecter la position des boules grâce à la vision par ordinateur et au Machine Learning.

3. Interface Utilisateur

Une interface Web est prévue pour permettre de visualiser le dessus du billard ainsi que de contrôler les actions du robot. Cette interface permet de superviser le jeu et de transmettre des ordres au robot. L'interface doit afficher en temps réel le robot et ses déplacements.

4. Serveur

Un serveur doit être développé afin de faire le lien entre l'interface web et le robot physique. Ce serveur doit garantir un échange de données rapide et en temps réel entre le système de commande et le robot et éviter le plus possible les pertes de données.

5. Simulateur

Un simulateur est prévu afin de tester les algorithmes que nous aurons développés sans avoir besoin d'utiliser le robot physique. Le simulateur devra prendre en compte les contraintes de latences possibles des websocket, afin de reproduire au mieux les conditions réelles. De plus, il sera nécessaire de reproduire au mieux les physiques de la réalité.

Étude de l’existant

1. Teutonic

Ce robot a été présenté par des chercheurs allemands lors de la conférence internationale de robotique et d'automatique en 2011. Il utilise la réalité augmentée pour jouer au billard et a réussi 80% de ses coups. Il commence par calculer un ensemble de coups possibles, puis choisit le coup à la difficulté la plus faible. Il anime ensuite ses bras de manière à viser et tirer de manière optimale pour réussir.

Teutonic, le joueur de billard | Shy Robotics

2. PR2 de Willow Garage

L’équipe de Willow Garage a programmé un robot PR2 afin qu’il puisse jouer au billard. Pour cela, il utilise une caméra haute-résolution et un programme d’intelligence artificielle pour localiser la table et les boules mais également pour pouvoir les suivre et calculer leur trajectoire.

Le robot PR2 joue au billard | Robot Blog

3. Deep Green

Ce projet du Robotics and Computer Vision Laboratory de la Queens University utilise des caméras se trouvant au-dessus de la table et un bras robotique pour jouer au billard. Le robot peut se déplacer sur des rails coulissants.

Deep Green, le robot qui joue au billard | Robot Blog

Ces exemples montrent que plusieurs équipes et institutions travaillent déjà sur des robots capables de jouer au billard. Ils utilisent des technologies similaires, telles que la vision par ordinateur, les marqueurs visuels et les algorithmes de machine learning. Cependant, on remarque aussi que chacun de ces exemples utilisent un robot avec un bras articulé tenant une canne, ce qui est très différent de notre projet avec un simple robot roulant.

Étude technique

1. Robot

Le robot possède une carte Arduino, qui servira à la fois à contrôler les mouvements du robot mais également à recevoir les requêtes du serveur.

a) Langage de programmation

Le code du robot sera développé en **C++** dans l'environnement Arduino IDE. Ce langage est natif pour Arduino et permet un contrôle efficace des moteurs.

b) Librairies

- **Wifi** : Cette bibliothèque permettra au robot de se connecter au réseau Wi-Fi. Elle gérera l'authentification et l'établissement de la connexion avec le point d'accès réseau, permettant une connexion stable au serveur.

- **ArduinoWebSockets** : Cette bibliothèque sera utilisée pour intégrer les WebSockets dans le code Arduino. Elle facilitera la réception et l'envoi des messages entre le serveur et le robot, permettant ainsi de gérer les commandes de déplacement en temps réel.

2. Interface web

L'interface utilisera les technologies du web afin de pouvoir à la fois faire un affichage esthétique des images prises par la caméra mais également de pouvoir envoyer des requêtes au serveur.

a) Langages de programmation

Pour le balisage et le style nous utiliserons **HTML / CSS**, ce sont les langages courants en développement web et n'ont pas vraiment d'alternatives. En ce qui concerne l'interactivité, nous avons choisi **Javascript** pour implémenter l'interactivité du site et les envois de requêtes au serveur, Javascript possède plusieurs bibliothèques dédiées à cela.

b) Librairies

- **OpenCV** : Librairie reconnaissant des objets grâce à un algorithme d'IA. La librairie nous sera utile pour reconnaître les boules de billard sur les images de la caméra.

- **JS-aruco** : Librairie détectant les tags Aruco. Elle nous permettra de repérer le robot et d'obtenir sa position et son orientation sur le billard grâce à son Aruco.

- **Socket.IO** : Librairie permettant l'envoie et la réception de websockets. Nous l'utiliserons pour envoyer des requêtes au serveur.

3. Serveur

a) Langage de programmation

Le serveur sera développé en **JavaScript** en utilisant **Node.js**. Nous faisons ce choix de langage et de plateforme pour la capacité de Node.js à gérer des opérations asynchrones et des connexions en temps réel, ce qui est nécessaire pour faire le lien entre l'interface web et le robot.

b) Librairies

- **Socket.IO** : Comme pour l'interface web, nous réutilisons cette librairie pour la réception et l'envoi de requêtes.

4. Simulateur

a) Langage de programmation

Comme dit plus tôt, le simulateur doit reproduire le plus fidèlement possible les conditions réelles. Nous allons donc utiliser le même langage que pour le serveur et l'interface web : **Javascript / HTML / CSS**.

b) Librairie

Il est important que le simulateur ait des physiques suffisamment réalistes, pour ce faire nous allons utiliser une librairies Javascript créée pour faire des physiques : **Matter.js**. Cette librairie permet de créer des objets et de leur ajouter des physiques (Gravité, Vélocité, Rotation, etc…) et de faire un affichage dans une balise canvas HTML.

Nous avons hésité avec la librairie Planck.js, mais Matter.js a une documentation beaucoup plus complète et est plus facile à prendre en main, c'est pourquoi notre choix s'est finalement porté sur Matter.js.

Problèmes pouvant être rencontrés :

La latence entre les requêtes des sockets : il faudra partir du principe qu'il y aura des latences à durées variables entre chaque instruction envoyée au robot, il faudra donc gérer cela pour que le système reste bien en temps réel, pour cela il faudra horodater les envois d'ordre pour que l'arduino sache lequel exécuter, tout ordre reçu avec trop de retard sera ignoré.

L'utilisation de la robotique est quelque chose de nouveau pour nous, il faudra également apprendre à utiliser le langage de l'arduino.

Il faudra que chacun participe à la conception et la réalisation de toutes les facettes du projet, plutôt que chacun soit spécialisé dans un domaine en particulier.

Faire tourner un robot : il faudra faire en sorte que le robot se déplace de façon fluide et non pas saccadée.