ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

par

Guillaume POIRRIER

RAPPORT DE PROJET PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE COMME EXIGENCE PARTIELLE À L'OBTENTION DE LA MAÎTRISE EN GÉNIE LOGICIEL

CONCEPTION DE L'INTELLIGENCE ARTIFICIELLE RÉGISSANT LE COMPORTEMENT D'UN ROBOT GARDIEN DE BUT DE SOCCER

MONTRÉAL, LE 10 DÉCEMBRE 2019

PRÉSENTATION DU JURY

CE RAPPORT DE PROJET A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

Professeur Alain April, directeur de projet
Département de génie logiciel à l'École de Technologie supérieure

Professeur Abdelaoued Gherbi, jury
Département de génie logiciel à l'École de Technologie supérieure

REMERCIEMENTS

Je tiens à remercier en premier lieu le professeur Alain April, pour la confiance qu'il m'a accordée en me confiant ce projet, ainsi que pour son accompagnement tout au long de mes recherches.

Mes remerciements s'adressent aussi à M. Mathieu Dupuis et M. Thierry Pouplier pour leur expertise en matière de Machine Learning.

CONCEPTION DE L'INTELLIGENCE ARTIFICIELLE RÉGISSANT LE COMPORTEMENT D'UN ROBOT GARDIEN DE BUT DE SOCCER

Guillaume POIRRIER

RÉSUMÉ

L'objectif de ce projet de recherche appliquée au niveau de la maîtrise en génie logiciel est d'établir la faisabilité de l'implantation d'un algorithme pour un gardien de but de soccer à l'aide de l'algorithme Q-Learning, dans le but d'améliorer la performance lors de la compétition Robocup. Le robot testé est un Nao version 6.

Le Q-Learning est une technique d'apprentissage machine par renforcement. Cela signifie qu'il octroie des récompenses, positives ou négatives, en fonction d'évènements détectés. Le poids de ces récompenses est ajusté afin d'obtenir un arbre de décision markovien. Cet arbre définit l'évolution des décisions prises, dans ce cas par le gardien de but, afin qu'il arrête la balle.

Dans un premier temps, une revue de littérature portant sur l'état de l'art en matière de Q-Learning et de son utilisation dans la Robocup est présentée. Ensuite, une proposition de solution théorique est proposée, comprenant les équations du comportement désiré du robot gardien de but, ainsi que les paramètres propres au Q-Learning. Deux modèles théoriques seront mis en compétition : un modèle dit « idéal » et un modèle dit « approché » concernant le point de convergence entre la trajectoire de la balle et du gardien. Le modèle idéal décrira une trajectoire selon deux axes et le modèle approché décrira un déplacement le long de la ligne de but. Un prototype expérimental est ensuite décrit et les résultats d'expérimentations sont présentés et interprétés, discutant de la performance et de la viabilité de cette approche pour l'amélioration de la performance future du gardien de but du club étudiant Naova de l'École de Technologie supérieure.

Mots-clés : Q-Learning, Nao, Robocup, Gardien de but, Robotique, Apprentissage machine, Apprentissage par renforcement, Arbre de décision markovien.

ARTIFICIAL INTELLIGENCE DESIGN GOVERNING THE BEHAVIOR OF A SOCCER GOALKEEPER HUMANOID ROBOT

Guillaume POIRRIER

ABSTRACT

The objective of this software engineering applied research project, at the master's level, is to establish the feasibility of implementing an algorithm for a soccer goalkeeper using the Q-Learning algorithm, in order to improve its performance during the Robocup competition. The robot tested is a Nao version 6, which has to behave as a goalkeeper for soccer.

Q-Learning is a reinforcement machine learning technique. This means that positive or negative rewards are awarded, based on detected events. The weight of these rewards is adjusted to obtain a Markovian decision tree. This tree defines the evolution of the decisions made, in this case by the goalkeeper, so that he prevents a goal.

First, a literature review of the state of the art in Q-Learning and its use in Robocup competitions is presented. Then, a theoretical solution proposal of is described, including the mathematical equations depicting the desired behavior of the robot goalkeeper, as well as the parameters specific to Q-Learning. Two theoretical models will be put in competition: a model called "ideal" and a model said "approximate" concentrating on a convergence point between the trajectory of the ball and the goalkeeper. The ideal model will describe a trajectory along two axes and the approximate model will describe a displacement along the goal line. An experimental prototype is then described and experimental results are presented and interpreted, discussing the performance and viability of this approach for improving the ÉTS Naova student club goalkeeper performance in the future.

Keywords: Q-Learning, Nao, Robocup, Goaltender, Robotics, Machine learning, Reinforcement learning, Markovian decision tree.

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

NAOVA	Club étudiant de robotique. Ce projet s'inscrit en collaboration avec celui-ci.
NAO	Robot humanoïde avec lequel la compétition de soccer se déroule.
SPL	Standard Platform League, qui est la ligue dans laquelle Naova est inscrite à la Robocup.

VARIABLES UTILISÉES

Nom de la variable	Description	Valeur
G	Position du gardien	Fournie dans les bibliothèques du Nao
I	Point d'intersection de la balle et de la ligne d'embut	Équation (1.2.1-1
L	Largeur du terrain	6000 mm
l	Largeur de la surface de réparation	2200 mm
α	Angle de tir	Équation (1.2.1-2)
A	Limite gauche de la zone d'en-but	Définie dans la réglementation
C	Limite droite de la zone d'en-but	Définie dans la réglementation
δ	Récompense si un but est encaissé	Déterminée avec les expérimentations
β	Récompense si le gardien se rapproche de I	Déterminée avec les expérimentations
γ	Si le gardien est placé sur la trajectoire de la balle allant au but	Déterminé avec les expérimentations
μ	Facteur d'apprentissage de la fonction Q	Déterminée avec les expérimentations, fixé à 0.9 par convention au début
γ	Facteur d'actualisation de la fonction Q	Déterminée avec les expérimentations, fixé par convention à 0.1 au début
$\emptyset$	Distance entre le gardien et la ligne de but, dans le cas du modèle approché	Déterminée avec les expérimentations

INTRODUCTION

Ce projet s'inscrit en collaboration avec un club étudiant de l'ÉTS, le club Naova. Ce club de robotique participe, depuis deux ans, à la compétition internationale Robocup dans la catégorie soccer. Cette compétition est reconnue mondialement pour ses avancées en matière d'Intelligence artificielle et est à la fine pointe mondiale en termes de recherche en vision, détection et anticipation du jeu par des robots Nao. Ce club étudiant participe aux compétitions de la Standard Platform League, aussi appelée SPL. Elle permet à deux équipes de robots Nao de s'affronter à cinq (5) contre cinq (5) sur un terrain de soccer spécialement dimensionné. Ce club est constitué d'étudiants en génie mécanique, électrique et logiciel qui se partagent mutuellement leurs compétences.

L'une des équipes les plus emblématiques et performantes de la compétition est l'équipe allemande B-Human. Le code source de Naova s'appuie sur leur version de 2016 qui a été utilisée/modifiée lors de ce projet de recherche.

CHAPITRE 1

REVUE LITTÉRAIRE ET ÉTUDE DU CODE EXISTANT

1.1 Introduction

Les systèmes intelligents font partie de la quatrième révolution industrielle, propre à ce siècle. Les avancées en matière d'intelligence artificielle sont appliquées dans des industries majeures telles que : le médical, les transports, le commerce, l'environnement et la Défense. Cette revue littéraire a pour but d'établir un état des lieux concernant les techniques populaires d'intelligence artificielle appliquée à l'apprentissage par renforcement et plus précisément le Q-Learning. Une attention plus particulière est apportée aux publications récentes de méthodes d'implémentation de systèmes intelligents sur des robots Nao, sur lesquelles porte ce projet de maîtrise appliquée. Enfin, les règles d'affaires de la Robocup y seront décrites, ainsi qu'un état des lieux du code source et de la documentation existante mise à disposition dans le club étudiant au moment de rédiger le présent rapport.

1.2 Apprentissage par renforcement

Cette spécialité particulière d'apprentissage machine consiste en l'application d'une fonction sur un agent évoluant dans un environnement afin de trouver la solution idéale à un problème. Les données utilisées pour l'autoapprentissage se font sur les données provenant de l'environnement réel du jeu. Bien sûr, il est aussi possible de simuler le jeu dans un environnement virtuel ou l'agent va apprendre à jouer avant de faire des essais réels avec le robot Nao lui-même. L'agent en question explore un arbre de possibilité et observe les réactions de l'environnement en fonction des valeurs des variables d'entrée. Il tente ensuite d'exploiter ces résultats afin de trouver la meilleure stratégie pour résoudre un problème. L'objectif final est d'établir un arbre de décision Markovien, c'est-à-dire d'appliquer un coût à chacune des actions de l'agent (c.-à-d. optimiser les récompenses versus les pénalités) afin d'optimiser la solution.

1.2.1 Théorie du Q-Learning

Le Q-Learning a d'abord été théorisé dans l'article *Learning from delayed rewards*, C.J.Watkins (1989) [1] et est une technique d'apprentissage par renforcement ayant la particularité de ne pas nécessiter de modèle défini d'environnement. Ses capacités universelles d'application en font un algorithme pouvant résoudre des problèmes complexes, avec un grand nombre de paramètres d'entrée. Cette particularité apporte une subtilité dans l'exécution de cet algorithme, à savoir la bonne définition des bornes des variables afin de pouvoir faire converger la solution au problème donné.

Les quatre (4) composants principaux d'une implémentation Q-Learning sont la matrice de valeur d'état, la fonction ou matrice de récompense, le facteur d'apprentissage et la fréquence d'actualisation. La matrice de valeur d'état est l'ensemble des valeurs des paramètres d'entrée. Le facteur d'apprentissage est une valeur $\alpha\epsilon]0,1]$ déterminant l'impact des nouvelles valeurs calculées sur les anciennes. Le facteur d'actualisation est défini selon $\gamma\epsilon[0,1]$ déterminant l'influence des valeurs des récompenses futures.

$$s = \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ \delta \\ \varepsilon \end{bmatrix} \qquad (1.2.1\text{-}1)$$

Figure 1 - Exemple de matrice d'état

Cette matrice décrit toutes les variables d'entrée souhaitées dans l'algorithme.

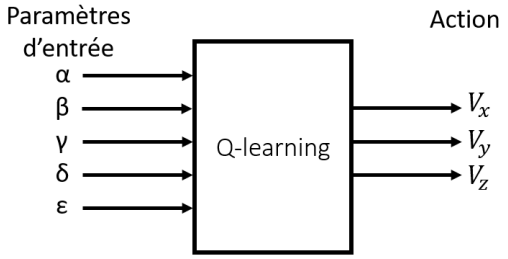


Figure 2 - Variables d'entrée d'un système en Q-Learning

Enfin, à chaque itération, les valeurs des Q valeurs sont déterminées en fonction de la fonction théorique suivante :

$$Q(s,a) = (1-\alpha).Q(s,a) + \alpha[R(s) + \gamma.max(Q(s',a'))]$$

Figure 3 - Fonction théorique du QLearning

(1.2.1-2)

1.2.2 Apprentissage du gardien de but avec Q-Learning

Le premier cas d'utilisation du robot gardien de but est l'arrêt de buts standard, c'est-à-dire dans une phase de jeu normale. Dans *Using Q-Learning to Control Robot Goal Keeper Behaviour*, Josha-David Fossel (2010) [2] sont présentées la gestion et l'interprétation des données d'entrée pour le robot Nao. Il explique qu'il souhaite établir une stratégie défensive spécifique au gardien de but à l'aide de la technique d'apprentissage par renforcement Q-Learning. Il tente de comparer l'efficacité d'un apprentissage machine et d'une stratégie codée en dur.

La stratégie codée est la suivante :

1. Le robot décide s'il doit agir en fonction de l'estimation du point d'impact en tenant compte des imprécisions de mesure. (2) est exécuté;
2. Si le point d'impact est dans la zone de but, le gardien décide d'agir. Il regarde la balle et tente de déterminer son accélération, pour savoir si elle va s'arrêter avant la surface de réparation, ou si elle peut entrer dans les buts. S'il est prévu que la balle entre dans la surface de réparation (3) est exécuté. Dans le cas contraire (1) est répété;
3. Le gardien de but tient sa position actuelle jusqu'à ce que le ballon soit dans la surface de réparation, si cela se produit dans les 4 secondes, le point (4) est exécuté. Dans le cas contraire (5) est exécuté;
4. Le gardien de but tente de dégager la balle. (5) est exécuté;

5. Le gardien se relève si besoin, et revient au centre des cages de but. (1) est exécuté de nouveau.

L'auteur conclut statistiquement que le gardien codé en dur est meilleur dans son cas, mais que les robots ayant appris en Q-Learning sont beaucoup plus adaptés au changement. Il précise aussi que dans des situations de plus en plus complexes, il devient irréalisable de s'appuyer sur du code en dur. Il conclut sa publication par l'énoncé qui propose, selon lui, que la technique de Q-Learning soit adaptée à l'apprentissage d'un gardien de but Nao.

1.2.3 Pénalités et Q-Learning

Le deuxième cas d'utilisation du robot est lors du tir de pénalités. Dans l'article *Making a robot stop a penalty*, Ruben van Heusden (2018) [3] publié par un professeur en intelligence artificielle de l'université d'Amsterdam, est décrite de quelle manière il est possible d'apprendre à un gardien de but Nao à arrêter un but à l'aide de la technique Deep Q-Learning. Cela signifie qu'il a utilisé un réseau de neurones pour post-traiter les valeurs de sortie de la fonction Q, et ainsi converger sur une solution plus rapidement.

La stratégie proposée dans cet article se base sur l'angle entre la balle par rapport à l'agent, en prenant comme point de référence le centre de la cage de but. L'agent reçoit une récompense positive en arrêtant un but, et reçoit une récompense négative en cas d'encaissement de but. Le principal problème observé par l'auteur, avec cette approche, est que l'agent pourra se rapprocher de la balle et effectuer une très grande partie du travail, tout en échouant. Il décrit comment il a donc fallu trouver un moyen de récompenser l'agent, même lorsqu'il n'a pas réussi à arrêter la balle, pour lui indiquer qu'il était en bon chemin de résoudre le problème. Pour contrer ce mode binaire de récompense, il propose un signal de récompense qui n'est pas basé seulement sur le mouvement de la balle, mais aussi sur la distance entre l'agent et la ligne représentant la trajectoire de la balle. Intuitivement, une récompense positive est attribuée lorsque la distance se réduit, et une récompense négative est infligée lorsque cette distance augmente.

1.3 Règles d'affaires de la compétition ROBOCUP

D'une manière générale, les règles de la Standard Platform League (SPL) sont semblables à du soccer traditionnel, avec un dimensionnement du terrain à échelle réduite et une adaptation aux capacités réelles des robots. Les parties sont donc de courtes durées pour s'ajuster à la lenteur des équipes ainsi qu'à la capacité de charge des Nao.

Les règles d'affaires spécifiques au gardien de but, décrites dans le règlement officiel de la Robocup [4], concernent surtout sa position sur le terrain à des phases de jeu telles que les penalties, la remise en jeu et les dégagements. Il faut impérativement que le gardien de but reste dans sa surface de réparation tout au long du jeu, sous peine de sévères sanctions par les arbitres. Il s'agit donc de bornes fixes avec lesquelles le robot va prendre des décisions.

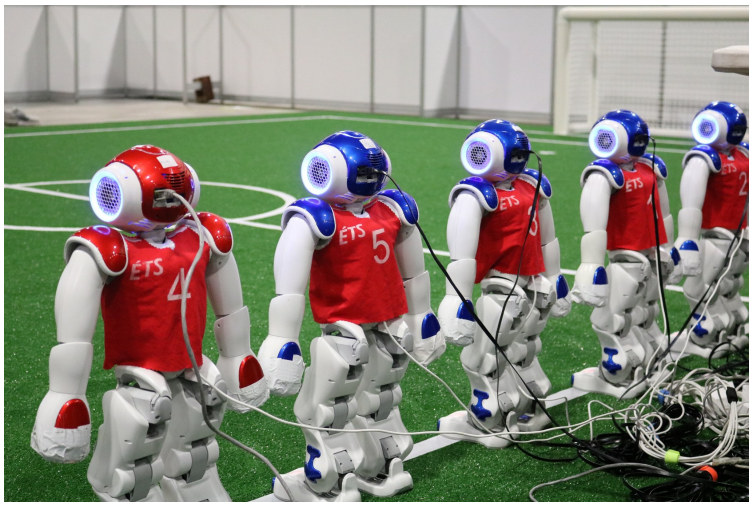
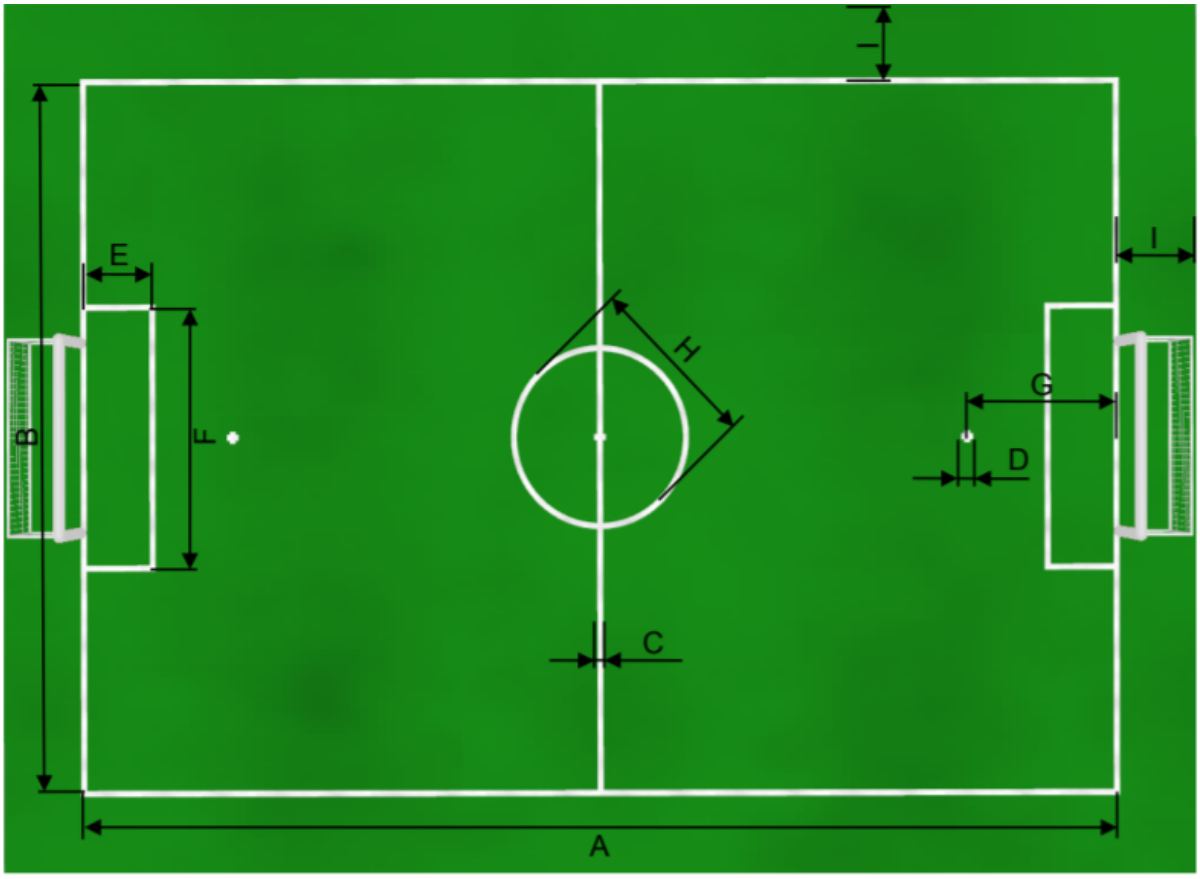


Figure 4 - Robots Nao en cours de configuration avant un match

Le terrain officiel, tel que décrit dans la réglementation de la Robocup, respecte les proportions standards du soccer, tout en étant d'une longueur de neuf (9) mètres. La surface de réparation, dans laquelle doit rester le robot, est clairement délimitée.



ID	Description	Length (in mm)	ID	Description	Length (in mm)
A	Field length	9000	E	Penalty area length	600
B	Field width	6000	F	Penalty area width	2200
C	Line width	50	G	Penalty cross distance	1300
D	Penalty cross size	100	H	Center circle diameter	1500
			I	Border strip width	700

Figure 5 - Dimensions officielles du terrain, tiré de RoboCup Standard Platform League (NAO) Rule Book (2019, p.2).

Au fil des expérimentations décrites plus loin dans ce rapport, il a été mis en évidence que la plupart des tirs cadrés de l'équipe adverse le sont au centre du but, c'est donc la zone la plus critique. Ce schéma est une visualisation de cette zone critique.

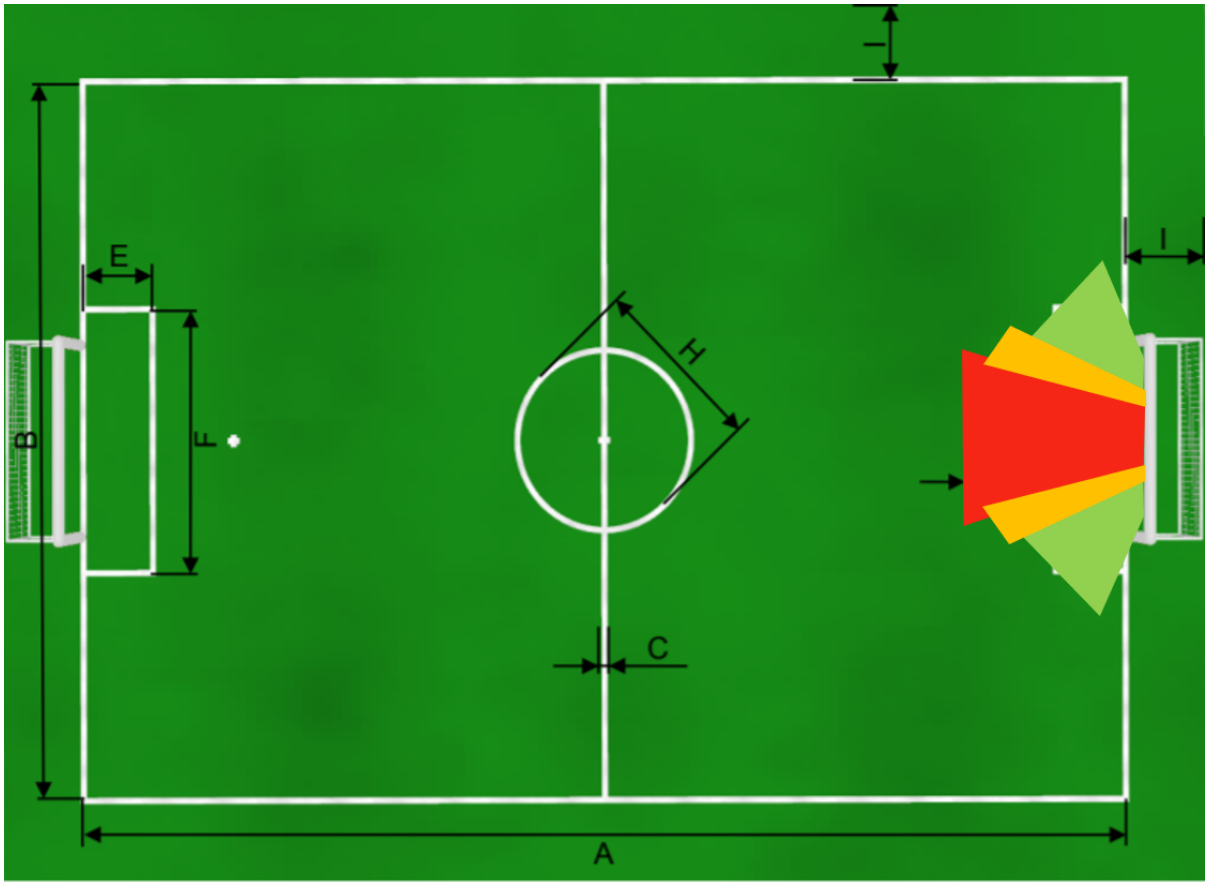


Figure 6 - Mise en évidence de la zone critique à défendre pour le gardien de but

La zone rouge indique le champ de provenance le plus probable des tirs adverses. Ce résultat est expliqué en section 2.3 de ce rapport.

1.4 Étude du code existant

Le code source, originalement délivré, par l'équipe allemande B-Human en 2016, est écrit en C/C++. Il comporte divers modules tels que la vision, les moteurs de déplacement ainsi qu'un logiciel complet de simulation. Les variables sur lesquelles s'appuiera le code se situent dans les modules concernés, et elles sont retranscrites pour la plupart dans la documentation fournie par B-Human. Aldebaran, la compagnie concevant et distribuant les robots Nao, propose un kit d'interface de programmation (SDK) spécialisé pour le robot, le NaoQi.

Le code de B-Human utilise ce SDK en plus de la mise en place de son propre cadriciel de développement logiciel. Une grande partie de la réalisation de ce projet a donc été consacrée à la compréhension et à l'exploration du code source actuel dans le but d'identifier les endroits où sera insérée l'intelligence articielle spécialisée pour un gardien de but.

Étant donné que cette recherche s'inscrit dans la continuité d'essais de mise en œuvre d'intelligence artificielle dans ce code source, il est important de bien saisir le détail des efforts effectués cette année par les autres équipes. La prochaine section décrit les différentes activités déjà réalisées récemment ainsi que leur essai d'implémentation, en commençant par l'environnement de simulation de l'équipe SimRobot. Viendront ensuite la description d'un module Nao ainsi que les expérimentations précédemment conduites au sein du club. Les conclusions et limites de ces implémentations sont aussi résumées.

1.4.1 SimRobot

Le SimRobot est un logiciel de simulation développé par B-Human. Il permet de simuler des scénarios en reproduisant le comportement de dix (10) joueurs. Il permet aussi de tester de manière empirique l'efficacité de l'apprentissage des robots. Il comporte cependant quelques limitations :

- Il n'est pas automatisé, donc à chaque but il faut reprendre la balle à la main (c'est-à-dire à la manière d'un « drag-and-drop ») pour recommencer la partie;
- Il s'exécute lentement, alors les robots se déplacent visuellement à la même vitesse que les vrais Nao.

Afin d'entraîner de manière efficace le comportement du robot, à l'aide d'algorithmes d'intelligence artificielle, il faudra trouver une solution de simulation plus rapide. Il existe une version « headless », c'est-à-dire sans affichage graphique de ce logiciel, mais le gain de performance observé en l'utilisant n'est que notoire.

Afin de tester une proposition d'algorithme de type Q-Learning, il sera nécessaire de modifier le logiciel de simulation afin qu'il capte, dans les fichiers de journalisation des évènements, les résultats sans affichage graphique. Cela permettra, de manière plus rapide, l'évaluation de la pertinence des modifications de code apportées. Thierry Pouplier, cofondateur et membre du club Naova, et moi-même travaillons actuellement à la modification de ce logiciel de simulation, afin qu'il puisse lui aussi implémenter un algorithme de drible à l'aide du Q-Learning.

1.4.2 Module Nao

Afin d'être qualifié à la Robocup, il faut être en mesure de montrer l'implémentation d'une fonctionnalité majeure dans le code de ses robots. Cela revient souvent à l'implémentation de ce que l'on appelle un nouveau module. Dans ce projet de recherche, nous allons donc réaliser un nouveau module nommé « QLearningKeeper », qui s'ajoutera aux modules déjà existants et spécialisés, par exemple, les modules : de vision, de gestion des moteurs, de gestion des déplacements et de la prédiction des trajectoires. En fonction des résultats obtenus en fin de projet, ce nouveau module pourrait permettre à l'équipe d'appuyer sa candidature à la compétition Robocup de 2020 s'il fait l'objet d'une publication scientifique.

1.4.3 Première tentative d'implémentation de Q-Learning au sein du club

Dans le cadre de son projet de fin d'études, Thierry Pouplier et son équipe ont tenté de faire l'apprentissage de la compétence de drible [5] au sein des robots Nao qui jouent un rôle offensif. Une première tentative de solution a été réalisée en C++, et par la suite ils ont dû changer de technologie et se sont tournés vers une tentative de solution en Python. D'après leurs observations, il devenait trop difficile d'interfacer le code existant du Nao avec les librairies C++ et le cadriciel logiciel créé par l'équipe B-Human. À la suite de cette première expérimentation en intelligence artificielle, leurs observations ont été les suivantes :

- L'apprentissage n'a pas abouti, car la fonction de Q-Learning ne convergeait pas;

- Cet apprentissage ne fonctionne pas, car il semble y avoir un trop grand nombre de paramètres en jeu pour un seul réseau de neurones;
- L'obsolescence de la simulation actuelle, livrée avec le code, n'a pas permis d'entraîner une grande quantité de générations à cause de sa lenteur d'exécution;
- La décision d'utiliser le langage Python n'a pas été un succès à cause de problèmes de performances et de problèmes d'interfaçage avec le cadriciel de B-Human.

Ainsi, pendant cette première expérimentation d'utilisation du Q-Learning, leur choix a donc été de retourner à nouveau vers l'utilisation du langage de programmation C++. Cette première expérimentation s'est achevée alors qu'ils tentaient d'exporter, en temps réel, les différentes variables de récompenses vers un fichier de format CSV, qui aurait permis de suivre l'évolution des entraînements successifs. En conclusion, cette équipe propose les pistes de solutions suivantes à titre de leçons apprises: « Suite à nos essais non concluants, nous croyons qu'un algorithme avec plus de paramètres (c.-à-d. ayant plus de couches) aurait plus de chance de converger vers un bon comportement. Des algorithmes tels qu'Actor-Critic ou Soft-Q-Learning seraient de bon candidat à être testé par la suite. Ceux-ci peuvent être décrits comme étant des algorithmes d'apprentissage profond. C'est-à-dire qu'ils contiennent plus réseaux de neurones à couche multiple. » [5], page 25.

Pour ce deuxième essai d'utilisation d'une technique d'intelligence artificielle (c.-à-d. ce projet de recherche appliquée), Thierry Pouplier s'est rendu disponible pour répondre à diverses de mes questions, et en sa qualité d'expert de l'implémentation du code au sein du club, il a soulevé certains points majeurs qui ont menés à l'échec de sa première tentative:

- Les robots sur lesquels les tests ont été effectués sont des Nao v5. Ces robots sont moins propices à l'apprentissage machine, car disposent de bien moins bonnes spécifications techniques (c.-à-d. puissance de calcul et mémoire disponible) que les v6;
- Le logiciel de simulation, tel qu'il existe actuellement et qui pose problème, sera remplacé par une nouvelle version dans les mois à venir. Il faut savoir que la version actuelle date de 1994;

- Le C++ n'est vraiment pas le langage de programmation idéal, à long terme, pour effectuer du Machine Learning, mais c'est une technologie qui peut être utilisée pour effectuer des preuves de concepts;
- Dans les deux prochaines années à venir, pour que le club puisse participer de nouveau à la compétition, ses membres devront créer de nouveaux modules et à terme totalement remplacer le bytecode actuellement utilisé qui provient de B-Human. Il est prévu que ces modules soient développés en Python, ce qui facilitera les implémentations futures en matière de Machine Learning.

1.5 Conclusion

La revue littéraire a énuméré les différentes solutions d'implémentation d'algorithme Q-Learning pour un robot Nao, tout en précisant les règles d'affaires régissant ce projet. Un des principaux enseignements retirés de l'analyse des publications du domaine est l'importance de l'utilisation d'un système de récompense non binaire, qui risque d'augmenter les risques de divergence du modèle. Une solution serait de prendre des décisions à l'aide d'un arbre de largeur n, qui prendrait en compte un nombre plus important de paramètres d'entrée. Il faudra aussi imposer, dans le code, des bornes pour respecter les règles d'affaires de la Robocup.

Ces publications ont aussi démontré qu'il est théoriquement faisable d'implémenter le Q-Learning dans un robot Nao et que ce genre d'implémentation a déjà été réalisé pour certains modules présentés à la Robocup. Finalement, la revue des essais antérieurs a aussi listé les difficultés à surmonter afin de réussir la mise en œuvre du Q-Leraning dans ce projet. Le prochain chapitre décrit la démarche expérimentale théorique de ce projet. Il tentera de tirer parti de cette revue littéraire ainsi que des observations des essais antérieurs.

CHAPITRE 2

PROPOSITION DE SOLUTION

2.1 Introduction

Les choix de solutions exprimés dans ce projet sont inspirés des enseignements provenant des cas décrits dans la revue littéraire, mais aussi des règles spécifiques à la Robocup. Les activités du projet qui ont été réalisées sont :

- L'étude approfondie du code existant et des règles d'affaires de la Robocup ;
- L'installation de l'environnement de travail et la conception d'un nouveau script d'installation qui permet aux nouveaux membres du club d'économiser un temps précieux lors de l'installation;
- Ce rapport, qui a fait l'objet de nombreuses révisions dès les premières semaines du projet, à raison d'environ une version par mois, selon l'avancement;
- Une stratégie et les calculs des bornes d'évolution pour les paramètres d'entrée de la matrice ont été définis ainsi que les équations de récompense;
- Une modification majeure visant la redéfinition de l'espace de liberté du robot, car auparavant il pouvait sortir de la surface de réparation.

Ce chapitre détaillera les différentes évolutions des livrables pendant le projet ainsi que les décisions techniques. Ensuite, une étude de statistiques, de matchs précédents de Naova pendant la Robocup est effectuée. À partir de ces travaux, une stratégie mise en place de l'algorithme de Q-Learning ainsi que la conception des équations régissant le comportement de gardien de but du robot sont proposées.

2.2 Évolutions de la démarche et livrables

Au fur et à mesure de la progression du projet de recherche, différentes approches ont été étudiées, notamment de compiler du code Python. Cette première piste de solution d'implémentation du Q-Learning est décrite plus bas et a été abandonnée en cours de projet.

Différents constats ont été faits au tout début de la collaboration avec le club Naova à la suite du démarrage de ce projet. Tout d'abord, le tutoriel d'installation des stations de travail du club souffrait d'obsolescence et était encore celui livré originalement par l'équipe B-Human en 2016. Ayant mis deux semaines à obtenir accès à un poste de travail fonctionnel, la première initiative de ce projet a été de concevoir, programmer et tester un nouveau script d'installation (voir Annexe II), afin de faciliter notre travail ainsi que celui des futurs collaborateurs du club.

À la suite de l'installation du logiciel, le premier objectif a été de comprendre le code source actuel du robot Nao. Le code actuel du gardien de but se contente de 2 actions : 1) repérer la balle; et 2) de se déplacer longitudinalement, dans la surface de réparation. À l'aide des données d'analyse de matchs, présentées à la section 2.3, il a été conclu que ces deux (2) actions n'étaient pas assez efficaces et coûtaient souvent la victoire à l'équipe lors de compétitions.

Étant donné les contraintes actuelles et les développements en cours au sein du club, ce projet de recherche appliquée, qui initialement visait à effectuer une implémentation et à un entraînement du gardien de but à l'aide de Q-Learning, il devient nécessaire de se réorienter vers une étude approfondie du code actuel incontournable et préalable qui vise à identifier comment insérer l'intelligence artificielle dans le code source actuel du Nao. En effet la taille importante du code actuel et les contraintes de la compétition contraignent le club à planifier ses activités d'amélioration des actions du robot sur plusieurs années afin de se qualifier. Les conditions de qualification de la Robocup imposent aux nouvelles équipes, telles que Naova (qui a fait sa première participation en 2018), à n'utiliser, et ce au bout de trois ans environ, que du code source qu'ils aient eux-mêmes conçu. Le club est donc forcé en ce moment à avoir

une bonne vue d'ensemble de l'architecture logicielle existante et à identifier une stratégie de remplacement de chaque composant logiciel. Une autre contrainte de la compétition est la possibilité/nécessité de collaborer avec d'autres équipes lors de matchs conjoints. La refonte de toute l'architecture logicielle et des modules devra aussi tenir compte, par exemple de l'interfaçage avec une autre équipe.

Conséquemment, ces contraintes, les travaux en cours ainsi que l'indisponibilité des robots Nao V6 réorientent la portée originale de ce projet à : 1) faire la synthèse de l'état de l'art; et 2) effectuer une étude de l'implémentation de la technique de Q-Learning dans le robot. Il ne sera donc pas possible d'effectuer une implémentation et un entraînement complet à cette étape. Les quatre livrables planifiés et effectués lors de ce projet de maîtrise appliquée seront donc :

1. Le code du module implémenté. L'intégralité des codes sources et de leur documentation se situe le répertoire du code Gitlab de Naova. Le code source du module se situe dans le sous-répertoire « Naova Code » et sous la branche « feature/qlearning-keeper »;
2. Le code et la documentation du script d'installation des environnements de développement de Naova. Le code source et la documentation du script d'installation des machines de développement de Naova se situent dans le sous-répertoire « Scripts » et se nomme « installation.sh »;

   La configuration d'une simulation dans le but de visualiser les effets de l'implémentation Q-Learning sur le gardien de but, Les scénarios de simulation des matchs se situent dans les fichiers de configuration du projet /Config/Scenes/, sous la forme des fichiers « QLearningKeeper.con » et « QLearningKeeper.ros2 ». Ces fichiers répondent à un standard XML en décrivant la position et le rôle des joueurs et sont directement interprétés par SimRobot.

2.3 Analyse préliminaire des matchs

Dans un premier temps, il a été nécessaire de concevoir une base de données contenant les données captées lors des matchs de la compétition. Ceci est utile afin de savoir ce à quoi le gardien de but est confronté lors de matchs. Cela permet aussi d'informer la recherche lors de la recherche d'une stratégie gagnante pour le robot gardien de but. Les données collectées sont très nombreuses. Conséquemment il est difficile lors de cette première recherche d'identifier les données qui ont un impact direct ou indirect sur la performance du gardien de but. Pour une première étude, les données suivantes semblent significatives :

- Le nombre de tirs cadrés;
- Le nombre de tirs non cadrés;
- La fréquence de réussite du gardien de but actuel;
- La vitesse moyenne et l'angle de la balle à arrêter;
- L'emplacement du franchissement de la balle dans les cages de but;
- Pour quelle tentative y a-t-il eu but.

Ces données (voir annexe 1) ne peuvent actuellement être collectées par déduction en visionnant des vidéos des matchs effectués par Naova les années précédentes. Elles permettent de visualiser le comportement, en situation réelle, des robots lors d'un match et corréler les données de la base de données collectées au même moment afin d'identifier une tendance ou une relation.

La zone de but a été divisée en 8 parties de 25 centimètres, selon la numérotation suivante :

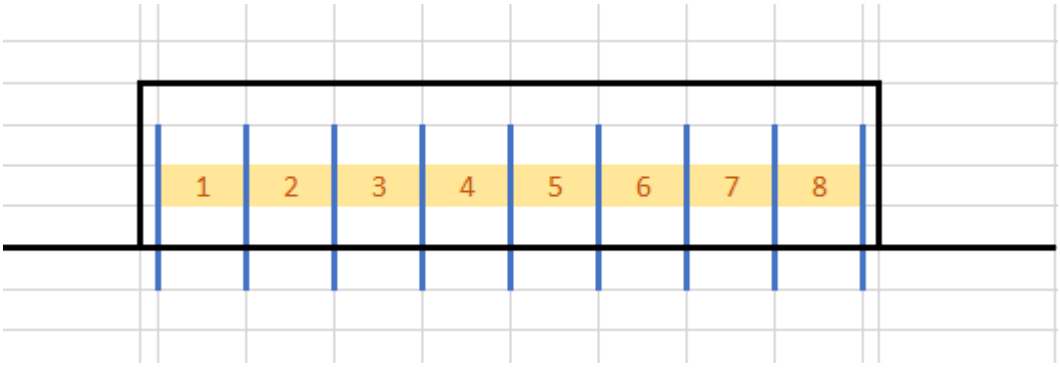


Figure 7 - Segmentation des zones de tir

Des gradients de couleur ont été appliqués permettant de voir apparaître la zone où le nombre de tirs cadrés, qui ont mené à un but, est le plus grand. On constate que les tirs aux extrémités du but sont bien plus rares que les tirs au centre (voir détails à l'Annexe I).

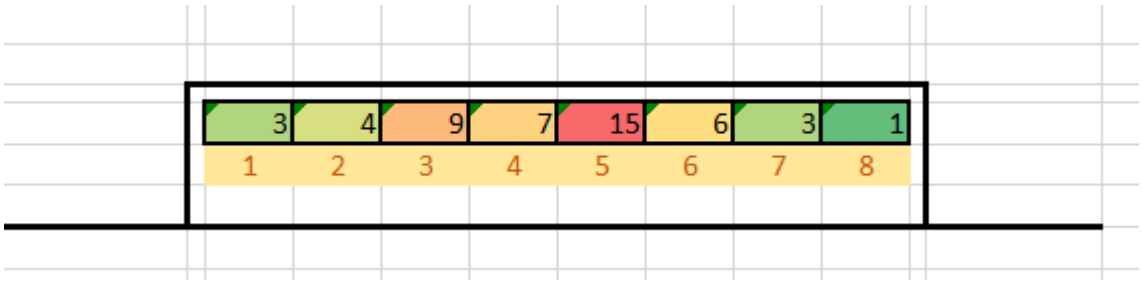


Figure 8 - Somme des tentatives de tirs cadrés associées à un gradient de couleur

D'autres données ont aussi été analysées et ainsi une synthèse de ces observations permet de constater les faits suivants :

- Les adversaires de Naova marquent en moyenne deux fois plus de buts qu'eux en match;
- Les coins du but ne sont que rarement visés.

D'autres observations ont aussi pu être faites à partir des données des matchs antérieurs. Il est considéré que le Nao est « perdu » s'il est en train d'essayer de trouver la balle, s'il est par terre sans raison ou s'il est en train de sortir des limites du terrain. Dans 62% des cas des buts

encaissés par Naova durant la compétition Robocup 2018, le gardien était considéré perdu au moment du but.

Nous avons tenté d'obtenir des données comportant la vitesse et l'angle des tirs provenant des adversaires, mais sans succès. Il est apparu trop complexe et imprécis de récupérer la vitesse moyenne des tirs à la simple observation des matchs. Le résultat de cette analyse permet cependant d'orienter les pistes d'amélioration possibles pour cette recherche et l'amélioration de la stratégie du robot gardien de but.

2.3.1 Stratégie de compétition

Lors d'une partie, il est observé que plus la balle est loin, moins elle risque de marquer un but, car elle offre un plus grand temps de réaction et un angle de tir plus faible. Les tirs les plus proches du but sont donc critiques, et une récompense supérieure devrait être accordée en fonction du point de départ du tir. Le comportement du robot Nao n'est pas comparable avec le comportement d'un gardien de but réel, car il lui semble vraiment difficile de ramasser une balle et de la renvoyer. Conséquemment, la stratégie proposée est donc simplement que le Nao doit apprendre à se déplacer afin de toujours se tenir entre la balle et le but. Quand cet apprentissage sera optimisé, il sera possible de penser à effectuer d'autres actions.

2.3.2 Décomposition des prises de décision

Dans cette recherche, il est établi que le comportement qui doit être appris se décompose en quatre (5) actions ou compétences suivantes:

- Diminuer l'intervalle de temps ou le gardien est perdu;
- Détecter rapidement la balle;
- Estimer adéquatement sa vitesse;
- Estimer justement sa trajectoire;

- Se déplacer rapidement entre la balle et le but (et y rester selon l'évolution du jeu).

Il faut cependant tenir compte du fait (c.-à-d., car il a été observé en regardant les vidéos pris lors de compétitions et en analysant les données historiques) que le robot peut aussi ne plus rien détecter pendant quelques dizaines de secondes. Selon les données disponibles et l'analyse effectuée à la section 2.3, 62% des buts sont encaissés alors que le gardien est perdu. Il faut donc primordialement que le robot apprenne à se positionner au point (c'est-à-dire à l'endroit dans sa zone de but) ou la plupart des buts sont encaissés historiquement. De cette manière nous augmenterons de beaucoup les chances d'arrêter la balle. Nous appelons ce fameux point, le point « X » dans le pseudo-code présenté à la figure 9.

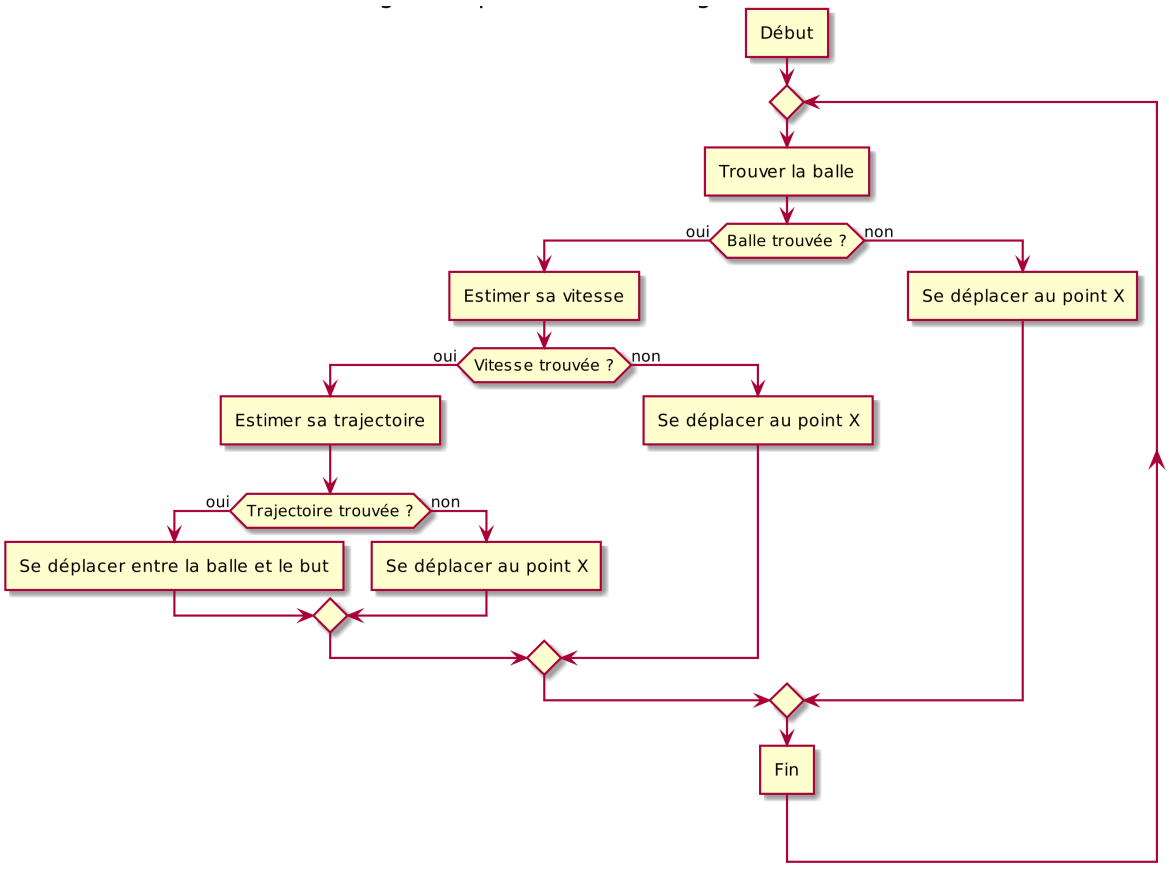


Figure 9 - Stratégie comportementale du gardien de but

2.4 Proposition de solution d'apprentissage par renforcement

Cette section décrit les paramètres pris en compte avec l'objectif de formuler une équation reflétant le comportement du robot gardien de but. Pour cela, les limites du terrain sont d'abord détaillées, puis le vecteur d'état et les équations des récompenses sont développés.

2.4.1 Référentiels sur le terrain

L'origine de la mesure présentée à la figure 10 se situe au le coin inférieur gauche du terrain. La ligne de but suit l'axe des X, tandis que la ligne de touche de gauche suit l'axe des Y. Le but a une largeur l, et le terrain une largeur L.

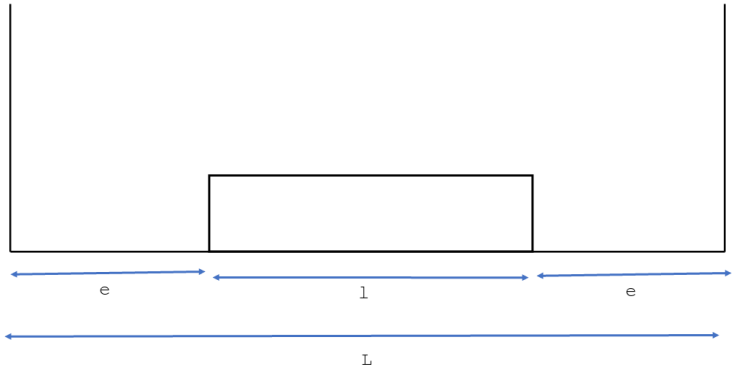


Figure 10 - Dimensionnement de la largeur du terrain.

Les dimensions officielles du terrain sont présentées au tableau 1.

Tableau 1 - Dimensions officielles du terrain

Description	Paramètre	Valeur (mm)
Largeur du terrain	L	6000
Largeur de la surface de réparation	l	2200

Ces dimensions constituent des limites absolues de la zone de déplacement du gardien, car il s'expose à de fortes pénalités s'il quitte sa surface de réparation.

2.4.2 Angle de tir α

L'angle de tir de la balle se nomme **α**, et sera pris en compte dans la fonction Q comme indiqué en section 2.4.5.

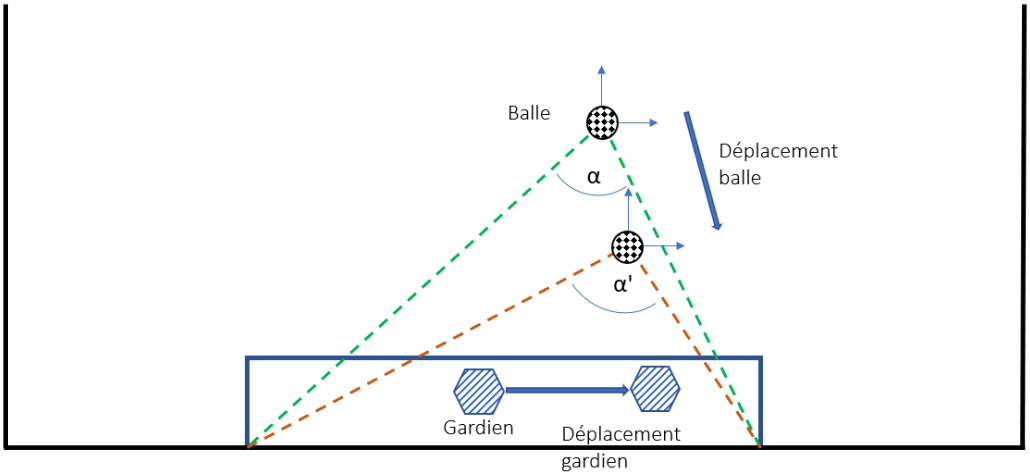


Figure 11 - Visualisation de l'angle de tir α

Plus cet angle est grand, plus la décision de déplacement du gardien devient critique, et donc plus les paramètres décisionnels vont être réduits pour offrir une réponse la plus rapide possible, même si elle est moins précise.

2.4.3 Définition des points de mesure

À la figure 12, la balle est située au point D et se déplace. Suite à ce déplacement, elle est détectée une nouvelle fois au point D'. L'espace entre les deux poteaux du but est localisé du point A au point C. I est le point où la balle va supposément entrer dans le but. B est le point d'intersection de la hauteur du triangle ADC sur le côté AC. Par analogie, le point B' est le point d'intersection de la hauteur dans le triangle AD'C sur ce même segment.

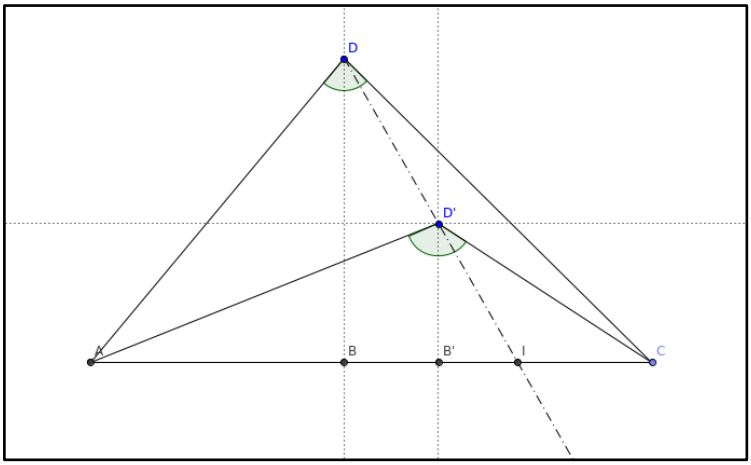


Figure 12 - Définition des points de mesure

2.4.4 Vecteur d'état

À l'aide des informations présentées dans ce chapitre, le vecteur d'état choisi pour tenter d'implémenter l'apprentissage par renforcement (c.-à-d. l'algorithme de Q-Leraning) est présenté à la figure 13.

$$S = \begin{bmatrix} Pg \\ Vg \\ Pb \\ Vb \\ \alpha \end{bmatrix} \qquad (2.4.4\text{-}1)$$

Figure 13 - Vecteur d'état

Avec :

P_g la position du gardien
V_g, la vitesse du gardien
P_b, la position de la balle
V_b, la vitesse de la balle
α, l'angle de tir de la balle

Les quatre (4) premiers paramètres de cette matrice sont directement fournis, en temps réel, par le SDK NaoQi. L'angle **α** est déterminé en fonction de ces paramètres, mais on

s'intéresse particulièrement ici à son évolution, car il semble que ce soit un facteur crucial de prise de décision, à savoir si le robot part en position sumo pour maximiser ses chances d'arrêter la balle. Cela consiste pour le robot à s'asseoir sur le terrain et à écarter les jambes.

Cette matrice représente les paramètres d'entrées de la fonction Q. Pour l'entraînement, ils seront générés aléatoirement afin de pouvoir affiner les paramètres de récompense, d'actualisation et d'apprentissage du robot.

2.4.5 Fonction de récompense

Le Q-Learning exige que l'on détermine une fonction de récompense (de l'anglais « reward function. ») afin de récompenser positivement ou négativement les actions du robot. Trois actions R_n ont été déterminées, ayant des récompenses $\delta, \beta \; et \; \gamma$. L'implémentation des récompenses se fait selon les règles suivantes :

$$R(S) = \begin{cases} \delta, & si\ un\ but\ est\ encaissé\ (R_1) \\ \beta, & si\ le\ gardien\ se\ rapproche\ du\ point\ I\ \ (R_2) \\ \gamma, & si\ le\ gardien\ est\ placé\ entre\ la\ balle\ et\ le\ but\ \ (R_3) \end{cases}$$

(2.4.5-1)

Figure 14 - Fonction de récompense

Les valeurs des récompenses seront déterminées au cours des séries d'entraînement du robot. Étant donné que les plus grands risques de buts sont lorsque la balle est tirée très proche de la ligne d'en-but, un bonus Θ est ajouté au gardien en fonction de l'ange de tir de départ de la balle.

$$\Theta = f(\alpha)$$

(2.4.5-2)

Nous obtenons ainsi :

$$R(S) = \begin{cases} R_1 = \delta, & avec\ \delta < 0 \\ R_2 = \beta, & avec\ \beta > 0 \\ R_3 = \gamma + f(\alpha) \end{cases}$$

(2.4.5-3)

Par application du théorème d'Al-Kashi [6], nous obtenons :

$$\text{Avec } \alpha = 180 - arctan\left(\frac{Yb-e}{Xb-e}\right) - arctan\left(\frac{Yb-e}{l-Xb-e}\right)$$

(2.4.5-4)

2.4.6 Détection des évènements

Les équations suivantes vont permettre au robot de déterminer les actions qui se produisent, et accorder ou non des récompenses.

R_1 - Détection de but

R_1 est délivrée si :

$$Yb < 0\ et\ Xb \in [e, e+l]$$

(2.4.6-1)

R_2 - Le gardien se rapproche du point I

Le point I est le point d'intersection de la trajectoire de la balle et de la ligne de but.

Coordonnées prévues de I :

$$I: \begin{cases} X_i = Xb + \dfrac{Yb.|Xb' - Xb|}{|Yb - Yb'|} + \dfrac{|L-l|}{2} \\ Y_i = 0 \end{cases}$$

(2.4.6-2)

Ce point possède une ordonnée nulle, car la ligne de but est située sur l'axe des Y. Lorsque le gardien se rapproche de ce point, la distance des points G et I diminue. Il est donc conclu que R_2 est délivrée si la valeur de N, la distance entre le gardien et I diminue selon l'équation :

$$N = \sqrt{(X_i - X_g)^2 + {Y_g}^2}$$

(2.4.6-3)

R_3 - Arrêt de la balle

Le point O est le point supposé idéal auquel le robot doit se diriger pour intercepter la balle.

Coordonnées du point de placement O :

Première solution : modèle idéal

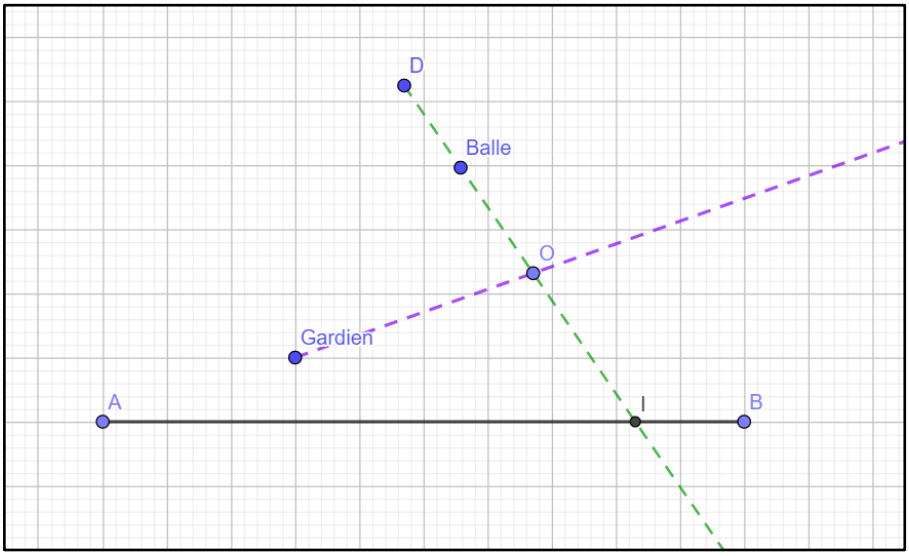


Figure 15 - Modèle idéal d'interception de la balle

Pour trouver le point O idéal, et afin que le gardien se place sur la trajectoire de la balle avant son passage, il faut satisfaire :

$$\frac{|GO|}{V_g} < \frac{|BO|}{V_b}$$

(2.4.6-4)

Ce qui revient à satisfaire :

$$\left|\left|\overrightarrow{V_g}\right|\right| \cdot \sqrt{(Y_g - Y_o)^2 + (X_g - X_o)^2} < \left|\left|\overrightarrow{V_b}\right|\right| . \sqrt{(Y_b - Y_o)^2 + (X_b - X_o)^2}$$

(2.4.6-5)

Et plus précisément :

$$\frac{\left|\left|\overrightarrow{V_g}\right|\right| \cdot \sqrt{(Y_g - Y_o)^2 + (X_g - X_o)^2}}{\left|\left|\overrightarrow{V_b}\right|\right| . \sqrt{(Y_b - Y_o)^2 + (X_b - X_o)^2}} < 1$$

(2.4.6-6)

<u>Deuxième solution : modèle approché</u>

Il peut être intéressant de déterminer une distance de la ligne de but à laquelle le robot doit toujours se tenir pour arrêter la balle. Il ne se déplacerait que selon l'axe X. Son point O aurait donc les coordonnées suivantes :

$$O : \begin{cases} X_O = X_i, & déplacement\ uniquement\ selon\ l'axe\ X \\ Y_O = \emptyset, & une\ distance\ entre\ la\ ligne\ de\ but\ et\ le\ gardien\ \text{à}\ déterminer. \end{cases}$$

(2.4.6-7)

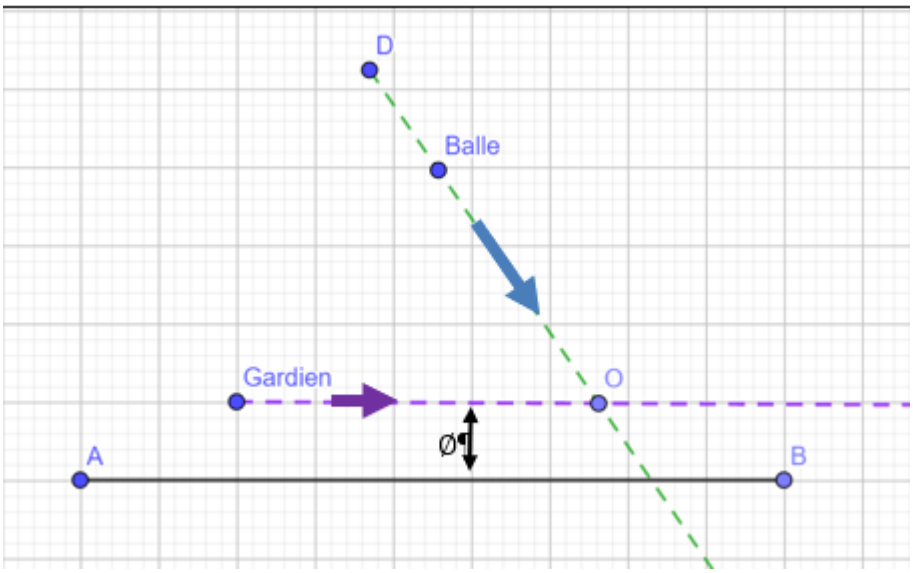


Figure 16 - Modèle approché d'interception de la balle

2.4.7 Détermination de la fonction Q

$$Q(s,a) = (1-\mu).Q(s,a) + \mu[R(s) + \gamma.max(Q(s',a'))]$$

(2.4.7-1)

Avec μ le facteur d'apprentissage et γ le facteur d'actualisation. Cette fonction sera déterminée au cours des expérimentations, ainsi que la valeur de tous les coefficients. Il est d'usage toutefois généralement de commencer par un facteur d'apprentissage à 0.9 et un facteur d'actualisation à 0.1, les premiers coefficients testés seront donc ceux-là.

2.5 Conclusion

Les deux modèles proposés ont été évalués et il est apparu rapidement que le modèle idéal ne tenait pas compte de la contrainte de la Robocup selon laquelle le gardien ne peut pas sortir de la surface de réparation, sous peine de pénalités. En effet, en ayant forcé le robot à ne pas franchir la ligne, ce modèle perd de son intérêt et offre des chemins bien moins optimisés

finalement que le modèle dit « approché », définit ci-dessus. Au début des expérimentations, les coefficients utilisés ont été les coefficients « standards », c'est-à-dire ceux conseillés dans la majorité des cas dans des problèmes de Q-Learning. Le chapitre suivant présente les travaux futurs envisagés à ce sujet.

CHAPITRE 3

Travaux complémentaires

3.1 Introduction

Les expérimentations ont été menées en utilisant les équations décrites précédemment dans l'environnement SimRobot. Ce chapitre du rapport présente les tentatives d'expérimentations effectuées ainsi que les recommandations quant aux futurs travaux à effectuer.

3.2 Module QLearningProvider

Les différentes implémentations impliquent la création :

- D'un « Role », KeeperLearner, qui définit la logique décrite en section 2.3.2 ;
- Un « Skill », DefendGoalLearning, qui va implémenter les fonctions propres au QLearning, et prendre des décisions. Ce skill est associé au « GetUp », qui est implémenté dans tous les rôles des joueurs et qui permet au robot de se relever en cas de chute.

Les fonctions utilisées pour récupérer les variables du jeu sont celles du SDK NaoQi, il n'a pas été nécessaire d'en créer de nouvelles. Ce code, écrit en C++, est supposé être exploitable par SimRobot, mais la simulation semble cassée. Après de multiples tentatives infructueuses pour modifier l'environnement, cette piste a été abandonnée. Il ne semble pas faisable d'exploiter la simulation telle quelle est, et d'entraîner un réseau de neurones.

3.3 Simulation de l'Université de Berkeley

Dans le but de mener à bien ce projet, et avec les conseils de M.Alain April et M.Mathieu Dupuis, j'ai suivi le cours *CS188 Intro to AI*, Dan Klein & Pieter Abbeel [7] concernant l'introduction au Q-Learning de l'Université de Berkeley. Ce cours a fait office de

propédeutique à ce projet dans le but de découvrir les grands concepts de l'apprentissage par renforcement. Il consiste à entraîner un agent dans le jeu Pacman. Une solution envisageable serait de s'émanciper définitivement de la simulation de B-Human et de se pencher sur la simulation offerte par ce cours. Elle contient l'ensemble des librairies Python sur mesure et un système d'entraînement complet. En mappant les données d'entrée de la matrice d'état du robot et en s'appuyant sur l'historique des matchs, il semblerait possible d'adapter cet environnement aux besoins de l'entraînement du Nao. L'idéal serait de pouvoir exécuter la simulation en mode « headless », pour offrir des performances maximales.

3.4 Analyse critique des résultats

Il n'a pas été possible d'obtenir des résultats exploitables avec la simulation par défaut. Il n'est pas possible de conclure que ce modèle théorique a conduit à une amélioration significative du comportement du gardien. Il faudra peut-être provoquer des actions plus aléatoires du robot afin d'obtenir de meilleures combinaisons de variables. Cependant cela augmenterait significativement le temps d'apprentissage, qui n'est déjà pas atteignable en l'état. D'après les conseils prodigués par M.Mathieu Dupuis, une solution pour résoudre ce problème pourrait être de s'appuyer sur les librairies d'entraînement SageMaker, d'Amazon Web Service. Cette suite possède un ensemble d'outils d'entraînement et offrirait une large capacité de computation, propice à un entraînement rapide.

3.5 Pistes d'amélioration

Pour maximiser les chances de convergence, ces différentes pistes doivent être explorées :

- La modification de la simulation afin d'accélérer le processus d'apprentissage ;
- La segmentation du terrain et des variables afin d'effectuer des calculs parallèles et ajouter des couches de réseaux de neurones, c'est-à-dire faire du Deep-QLearning ;
- Redéfinir les bornes et limites des variables étudiées afin d'optimiser le temps d'apprentissage et de réduire les problèmes liés aux extrémums locaux.

- L’adaptation de l’environnement de simulation de l’Université de Berkeley afin de ne plus être obligé de s’appuyer sur du code en C++ et le framework développé par B-Human.

Le club souhaite mettre à jour la simulation actuelle. Les avancées actuelles et les besoins grandissant de Machine Learning dans le cadre de la compétition auront peut-être conduit les développeurs de SimRobot à faciliter l’entraînement de réseaux de neurones dans cette nouvelle mouture.

CONCLUSION

L'étude du comportement des robots Nao et la revue littéraire ont permis d'affiner le modèle théorique proposé. Les essais sur le terrain et les difficultés liés à la simulation ne permettent cependant pas de conclure quant à l'efficacité réelle de l'approche pour le moment. Une des possibilités envisagées serait d'affiner les résultats avec un algorithme de back-propagation de type A* lié à un ajout de couches successives de réseaux de neurones et de segmentation du terrain qui permettrait peut-être de faire converger plus rapidement la solution. En attendant qu'un environnement de simulation stable soit disponible, l'application d'une simulation théorique pourrait se faire dans l'environnement de test du cours de l'Université de Berkeley (c.-à-d. le code de PacMan), ou tenter d'utiliser la bibliothèque SageMaker mise à disposition par Amazon Web Service.

Cette recherche apermis d'explorer des pistes de solutions pour de futurs travaux du club Naova dans sa recherche d'implémentation de Machine Learning pour son gardien de but.

ANNEXE I

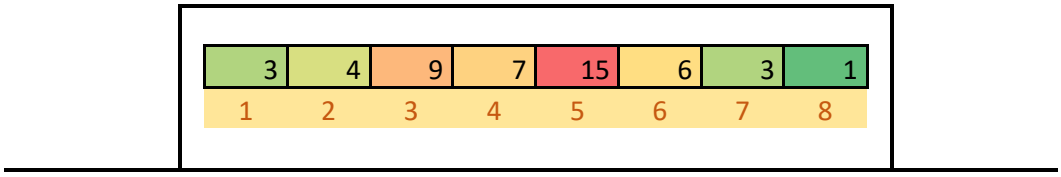
Étude des matchs existants

Scores et moyenne de buts par match en Robocup 2018

Naova	Adversaire
1	2
3	1
1	4
2	0
0	0
0	6
1	0
0	0
0	2
0,88888889	1,66666667

Analyse des zones de tentatives de tir pour 5 matchs de la Robocup :

		Tentatives de tirs par zones de pénétration dans le but						
		Matchs Robocup 2018						Total
Zone	1	1	0	0	0	1	1	**3**
	2	0	1	1	1	1	0	**4**
	3	3	0	3	1	2	0	**9**
	4	0	1	2	1	3	0	**7**
	5	3	3	1	3	2	3	**15**
	6	1	0	0	2	2	1	**6**
	7	0	1	1	0	0	1	**3**
	8	0	1	0	0	0	0	**1**



ANNEXE II

Script d'installation du projet Naova

La documentation de ce script se trouve dans le répertoir GitLab de Naova.

```
#!/bin/bash
#
# Script for naova installation
# Author: Guillaume POIRRIER
#
echo "WELCOME TO NAOVA !"
sudo apt update -y
sudo apt install -y git
echo "Code will be stored in \"~/naova\" directory by default"
NAOVA_ROOT=~/naova
read -p "User Git mail :"  usermail
git config --global user.email "$usermail"
read -p "Naova Gitlab username :" username
git config --global user.name "$username"
if ! [ -f $NAOVA_ROOT ]
then
        git clone https://gitlab.com/ClubNaova/NaovaCode.git/
$NAOVA_ROOT
fi
echo "Installing libraries..."
sudo apt-get install -y clang-4.0 qtbase5-dev libqt5svg5-dev libglew-
dev libxml2-dev graphviz xterm make
echo "Downloading Naoqi SDK..."
cd $NAOVA_ROOT
wget http://23.95.228.106/naoqi-sdk-2.1.4.13-linux32.tar.gz
echo "Decompressing NaoQi while executing Alcommon..."
./Install/installAlcommon naoqi-sdk-2.1.4.13-linux32.tar.gz
rm -rf naoqi-sdk-2.1.4.13-linux32.tar.gz
while true; do
    read -p "There we go ! ready to build the project ? [Y/n]" yn
    case $yn in
        [Yy]* ) echo "V--COMPILING--V"
                cd $NAOVA_ROOT/Make/Linux;
                make;
                break
                ;;
        [Nn]* ) break;;
        * ) echo "Please answer yes or no.";;
    esac
done
```

ANNEXE III

Fichier de configuration SimRobot

```
<Simulation>

  <Include href="Includes/NaoV4H21.rsi2"/>
  <Include href="Includes/Ball2016SPL.rsi2"/>
  <Include href="Includes/Field2015SPL.rsi2"/>

  <Scene name="RoboCup" controller="SimulatedNao" stepLength="0.01"
color="rgb(65%, 65%, 70%)" ERP="0.8" CFM="0.001" contactSoftERP="0.2"
contactSoftCFM="0.005">
    <!-- <QuickSolver iterations="100" skip="2"/>  -->
    <Light z="9m" ambientColor="rgb(50%, 50%, 50%)"/>

    <Compound name="teamcolors">
        <Appearance name="black"/>
        <Appearance name="red"/>
    </Compound>

    <Compound name="robots">
      <Body ref="Nao" name="robot1">
        <Translation x="0.9" z="320mm"/>
        <Rotation z="180degree"/>
      </Body>
    </Compound>
    <Compound name="balls">
      <Body ref="ball">
        <Translation z="1m"/>
      </Body>
    </Compound>

    <Compound name="field">
      <Compound ref="field"/>
    </Compound>
  </Scene>
</Simulation>
```

LISTE DE RÉFÉRENCES BIBLIOGRAPHIQUES

[1] Watkins, Christopher, 1989. « Learning From Delayed Rewards ». Thèse de doctorat, King's College, 241 p.

[2] Fossel, Joscha-David. 2010. « Using Q-learning to Control Robot Goal Keeper Behaviour ». Article scientifique, 9p.

[3] van Heusden, Ruben. 2018. « Making a robot stop a penalty ». Thèse de doctorat, Université d'Amsterdam, 24 p.

[4] Robocup Technical Committee. 2019. « RoboCup Standard Platform League (NAO) Rule Book ». p.2-4 et 14-22.

[5] Pouplier, Thierry, Brice Brouseau-Rigaudie et Marc Antoine Dumont. 2018. « Apprentissage par renforcement appliqué à des robots bipèdes », Projet de fin d'étude de baccalauréat, École de Technologie Supérieure, 27 p.

[6] Al-Kashi. XVe s. « Théorème d'Al-Kashi »

[7] Klein, Dan, Pieter Abbeel. « CS188 Intro to AI ». <http://ai.berkeley.edu/home.html>