MiRoCo un simulateur pour systèmes multi-robots à forte dynamique d'interaction

Lounis Adouane

LASMEA, UMR CNRS 6602
24 Avenue des landais, 63177 Aubière Cedex, France
Lounis.Adouane@lasmea.univ-bpclermont.fr

RÉSUMÉ. *Cet article a pour objectif de mettre en exergue les principaux éléments de conception et de modélisation liés à une simulation rigoureuse de systèmes multi-robots. Ces différentes modélisations vont être détaillés, discutés et analysés afin de permettre la mise en place du simulateur MiRoCo (MiniRobotique Collective). Ce simulateur est dédié à l'étude des systèmes multi-robots à forte dynamique d'interaction, plus spécifiquement, il permet de vérifier en condition quasi réelles les architectures de contrôle/commande proposées pour la coopération de robots mobiles communiquants. En effet, pour permettre de tirer des conclusions fiables sur ces structures de contrôle/commande, il est primordial que les modèles implémentés en simulation puisse tenir compte d'un ensemble de contraintes, les plus importantes correspondant aux contraintes temporelles et physiques du système étudié. Les premières influencent l'évolution globale du système multi-robots et les secondes caractérisent entre autres les interactions locales entre robots. MiRoCo permet actuellement des simulations exhaustives de plusieurs tâches génériques pour systèmes multi-robots.*

ABSTRACT.

MOTS-CLÉS : *Simulation multi-robots, Coopération de robots mobiles, Intelligence Artificielle Distribuée, Conception multi-agents, Modélisation UML.*

KEYWORDS: *MiRoCo simulator, Cooperative mobile robotics, Distributed Artificial Intelligence, Multi-robots simulation, Multi-agents conception, UML modeling.*

e soumission à , le .

1. Introduction

Le besoin de validation par simulation s'avère être un passage obligé avant toute phase de mise en oeuvre effective des solutions proposées pour le contrôle de systèmes complexes. Le contrôle réactif d'un groupe de robots mobiles fait partie de ces systèmes complexes, très difficiles à modéliser donc aussi à contrôler. En effet, le fait de ne pas disposer de modèles qui puissent décrire de manière précise le fonctionnement des systèmes multi-robots, fait que la plupart des travaux effectués dans ce cadre se basent majoritairement : soit sur l'évaluation de métriques caractérisant la réalisation des tâches coopératives à effectuer [RAM 94], [PAR 01] ; soit sur l'observation d'apparition de patterns dans l'environnement [MEL 01] ; ou bien sur des études statistiques de l'influence de tel ou de tel paramètre sur le bon fonctionnement des tâches à exécuter [BAL 95b], [KUB 97b], [ADO 04b].

Le fonctionnement d'un système multi-robots est principalement lié au type de contrôle qu'on lui applique. Cependant, il est aussi soumis à une multitude d'autres types de paramètres hétérogènes. Ces paramètres sont liés aux phénomènes physiques tels que : les frottement ou les glissements, les interférences des signaux lors des communications, les aléas de fonctionnement des capteurs. Tous ces paramètres réunis font que les phases de validation expérimentale sur les systèmes multi-robots deviennent longues et fastidieuses. En effet, il n'est pas toujours aisé de savoir quelle est l'incidence de tel ou tel paramètre sur les dynamiques d'évolution atteintes par le système.

Dans le but de réduire le temps consacré à ces phases de validation expérimentale, on est bien souvent contraint d'isoler les phénomènes (par groupe ou individuellement) afin d'étudier quelle est leur véritable influence sur le système complexe étudié. Dans ce cadre, *l'étude par simulation* s'avère être d'une importance capitale pour l'observation et l'explication des phénomènes tels que les comportements émergents liés à l'intelligence collective. Les propriétés formelles et les paramètres induisant ces phénomènes sont ainsi isolés et permettent par conséquent d'explorer les possibilités effectives du contrôle proposé, et ce, indépendamment des contraintes matérielles qui peuvent être trop particulières, et fausser ainsi l'analyse induite sur les contrôles proposés.

Les autres avantages liés aux phases de simulation sont nombreux et variés, nous citons par exemple :

– le gain en temps durant les phases de développement et de test,

– l'évitement du problème des pannes inopinées des robots réels (e.g., une soudure qui rompt, un composant défectueux, une alimentation faible),

– l'exhaustivité des possibilités de test de tout type de paramètres conditionnant les systèmes multi-robots, comme par exemple l'étude de l'influence du paramètre *nombre* de robots participant à la tâche coopérative, c'est-à-dire l'étude de l'effet de masse. Effectivement, ce paramètre ne peut pas être étudié d'une manière exhaustive en expérimentation matérielle, à part si on se restreint à un nombre de robots très réduit (de l'ordre de la dizaine), et même dans ce cas de figure, les résultats seront très

dépendants des aléas caractérisant le système multi-robots,

– la réduction des coûts dans le cas des systèmes trop onéreux pour procéder à de multiples phases d'essais-erreurs.

Les points cités ci-dessus et bien d'autres nous ont confortés dans l'idée de passer par des phases de simulations intensives (i.e., en mode *batch*) pour valider des points clés liés aux architectures de contrôle proposées dans [ADO 04a], [ADO 04b] et [ADO 05b].

Le développement du simulateur *MiRoCo* "*MiniRobotique Collective*" a été l'occasion pour nous de concevoir un environnement multi-agents, avec toutes les techniques de conception, de modélisation et de programmation que cela impose. Le développement de *MiRoCo* a été principalement motivé par le fait de disposer d'un outil qui modélise le plus fidèlement possible ce que nous voulions réellement simuler, en l'occurrence un groupe de robots qui puissent communiquer et interagir d'une manière précise, et ceci afin de réaliser des tâches de coopération. Une fois obtenues les spécificités souhaitées de *MiRoCo*, les tests des architectures de contrôle proposées dans [ADO 05a] se sont faits d'une manière toute naturelle, et ceci en gardant à l'esprit que les dynamiques d'interaction générées entre agents dans *MiRoCo* tendent à représenter le plus fidèlement possible le modèle réel du système multi-robots.

Cet article ne se veut pas exhaustif par rapport aux détails techniques du développement de *MiRoCo*, mais plutôt un article traitant de certaines spécificités conceptuelles et de modélisation d'un environnement multi-robots. Ces spécificités conditionnent le fait que le modèle simulé se rapproche ou non du système multi-robots réel.

La suite de cet article est organisé comme suit. La section 2 présente succinctement l'utilité des tâches génériques dans le cadre de la robotique mobile coopérative. La section 3 est un état de l'art sur les simulateurs multi-robots. Les sections 4, 5, 6 et 7 donnent par la suite les détails de conception et de modélisation du simulateur *MiRoCo*. Cet article se termine par des conclusions et perspectives.

2. Robotique coopérative et tâches génériques

Les tâches génériques en robotique collective ont comme principal objectif de fournir des tâches de référence "*benchmark*" pour évaluer les architectures de contrôle proposées dans la littérature. Ces tâches génériques sont censées exciter l'ensemble des modes de fonctionnement du système multi-robots. Par conséquent, la viabilité, la pertinence et les différentes caractéristiques qu'offre le contrôle utilisé sont ainsi systématiquement évaluées. Parmi les tâches génériques couramment utilisées dans la littérature nous citons : le *mouvement en formation* [PRE 90], [WAN 91], [CHE 94], [BAL 95a], [YAM 01], le *fourragement* [STE 90], [ARK 92], [DRO 93], [MAT 94], [BAL 95b], [SIM 01], ou la Tâche Coopérative de Poussée d'Objets TCPO [RUS 95], [MüL 98], [KUB 00], [BAL 03], [MUñ 03], etc. Cette dernière tâche coopérative nous a servi principalement comme base d'essai des différentes possibilités liées aux archi-

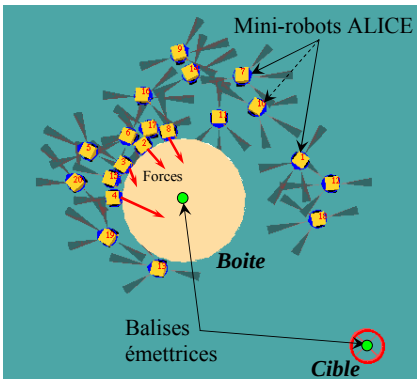


Figure 1. *Tâche coopérative de poussée d'objets "TCPO"*

tectures de contrôle que nous avons proposées [ADO 05a]. L'objectif général de cette tâche est de pouvoir déplacer d'une manière coopérative un objet d'une position initiale vers une destination finale. L'objet à pousser est choisi de telle sorte que son poids et/ou ses dimensions soient largement plus élevés que ceux d'un robot élémentaire. La coopération dans ce cas de figure est donc impérative (cf. figure. 1).

3. Etat de l'art sur les simulateurs multi-robots

Simuler un phénomène, un système, ou tout autre élément modélisable, pose toujours la question importante du niveau de détail à atteindre pour que les simulations soient satisfaisantes. Même s'il est évident que le monde est sans nulle doute son meilleur modèle[1], il reste qu'il y a toujours des phénomènes physiques plus prédominants que les autres par rapport à l'objet précis des simulations. C'est ce qui nous permet donc de nous restreindre à la modélisation des phénomènes importants. Cependant, même dans ce cas de figure, il faut généralement se restreindre aussi à un niveau de détail qui satisfasse le dilemme temps-précision[2]. En effet, en plus du fait qu'une modélisation précise des phénomènes ralentit considérablement les simulations, il y a aussi le fait qu'il n'est pas nécessaire d'atteindre des niveaux de précision très importants pour pouvoir tirer des conclusions ou valider les méthodes, les concepts ou les solutions apportées.

L'investigation menée au début de nos travaux de recherche sur les différents environnements de simulation que nous pouvions potentiellement utiliser, a montré l'existence de plusieurs types d'outils de simulation pour systèmes multi-robots (ou plus généralement pour systèmes multi-agents *situés* [SIM 01]). Certains outils relative-

1. "*The World is its own Best Model*" [BRO 90].
2. Par temps-précision nous voulons dire, temps nécessaire pour obtenir les résultats et précision souhaitée de ces derniers.

ment plus basiques simulent des environnements discrets[3], avec des robots à nombre d'états ou de perceptions réduit. D'autres outils plus sophistiqués vont jusqu'à la modélisation de la physique de l'environnement dans lequel évoluent les différents robots. Le choix du degré de sophistication du simulateur à utiliser dépend de l'objet même des simulations. En effet, le simulateur utilisé pour valider des concepts ou des mécanismes organisationnels d'entités autonomes ne sera pas forcément le même que celui qui sera utilisé pour l'étude des interactions microscopiques entre entités autonomes situées.

Parmi les environnements de simulation qui restreignent les positions des robots à des positions discrètes dans l'environnement (correspondant aux cases d'un cadre figé de l'environnement), nous pouvons citer le simulateur Alia (cf. figure. 2(a)) de Claude Delaye [DEL 93] qui simule des robots explorateurs chargés de récolter du minerai sur une planète inconnue. Le but est d'étudier, par le biais de l'évolution artificielle, l'adéquation matérielle et de contrôle de la colonie de robots utilisée pour réaliser la tâche en question. Une fois le minerai trouvé, il est rapporté à la base. Le cheminement de retour est facilité par la dépose à l'aller de "balises" (cf. figure. 2(a)). La particularité de ce travail réside dans le fait que les robots explorateurs sont générés automatiquement par la base, suivant un processus de sélection, basé sur leur capacité à rapporter du minerai. Les mutations, que peuvent subir les robots d'une génération à l'autre, sont à la fois comportementales (variation du contrôle qui leur est affecté), mais également morphologiques (comme l'utilisation de roues ou de chenilles, taille de la benne, etc.) et ceci en fonction de la nature supposée du terrain où évoluent les robots.

Un autre exemple de simulateurs utilisant un nombre d'états discrets, est celui développé et utilisé par Alexis Drogoul [DRO 93] dans le cadre du projet MANTA "*Modeling an Anthil Activity*" au LIP6. Ce simulateur (cf. figure. 2(b)) est dédié à l'étude des sociétés de fourmis en se basant sur le modèle de comportement individuel EMF "*EthoModelling Framework*". MANTA a permis de tester des hypothèses concernant l'émergence de structures sociales, telles que la division du travail, la sociogénèse, les relations de dominance et de hiérarchie au sein d'un type particulier de colonies de fourmis (Ectatomma ruidum). L'univers simulé correspond donc à une fourmilière artificielle, telle qu'on peut la voir dans un laboratoire d'éthologie. Chaque fourmi est représentée par un agent pouvant se déplacer de case en case et/ou effectuer des tâches particulières (soins aux oeufs, fourragement, etc.) suivant le principe d'EMF proposé. MANTA offre des possibilités importantes pour spécifier aisément les comportements des agents, ainsi que de nombreuses facilités graphiques. Cependant, cette plate-forme est plus particulièrement dédiée à la représentation et la validation de concepts organisationnels plutôt qu'à la simulation des interactions physiques précises entre individus situés dans un environnement contraint.

3. La discrétisation de l'espace d'état réduit considérablement les dynamiques d'interaction et d'évolution que l'on peut observer. En effet, ceci entraîne par exemple une représentation partielle des états atteignables par le système multi-robots.

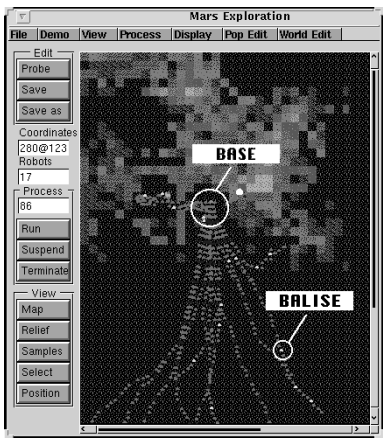
La communauté travaillant sur les SMA, dispose d'un grand nombre d'outils de simulation traitant de problèmes complexes via une modélisation par SMA. Pour le cas des simulations de systèmes multi-robots, leur transposition aux plate-formes SMA déjà existantes reste quasi immédiate vu la nature déjà distribuée du système multi-robots. Cependant, la plupart des plates-formes SMA offre pas ou peu de composants dédiés pour manipuler efficacement l'évolution d'agents situés dans un environnement contraint. Olivier Simonin par exemple dans [SIM 01] a développé un outil de simulation (cf. figure. 2(c)) qui s'exécute sur Madkit [4] [GUT 00], mais il a dû développer au préalable une librairie de classes spécialisées qu'il a interfacé avec les classes abstraites définies dans Madkit. Madkit a permis entre autres de gérer et de tester librement l'ordre d'exécution des actions des agents dans l'environnement, de développer facilement plusieurs représentations graphiques de l'exécution d'une même simulation et d'intégrer des sondes affichant des informations sur le système. Cependant, les actions des agents présents dans les simulations (cas des robots découpeurs ou pousseurs par exemple) restent à un niveau d'abstraction assez élevé par rapport à la physique de manipulation tels que la poussée ou le découpage des objets présents dans l'environnement.

Dans le cadre des simulateurs dédiés pour réaliser la tâche coopérative de poussée d'objets, on trouve aussi "SimbotCity" (cf. figure. 2(d)) développé par Ronald Kube [KUB 97a] durant sa thèse de doctorat afin de simuler entre autres les déplacements d'un objet poussé sous l'effet des actions d'un groupe de robots réactifs [KUB 94]. On note que dans le cadre de SimbotCity c'est un modèle géométrique qui décrit la relation existant entre les actions des robots et les déplacements consécutifs de l'objet à pousser.

Les simulateurs dédiés pour les robots footballeurs constituent d'autres types de plate-formes de simulation partagées par plusieurs équipes de recherche de par le monde. Ces équipes traitent généralement des problématiques voisines de celles du contrôle coopératif d'un groupe de robots mobiles. Dans le cadre de la fédération de robots footballeurs "RoboCup" par exemple, il y a en plus des compétitions de véritables robots mobiles, une autre ligue qui se focalise uniquement sur des compétitions en simulation. Le simulateur nommé "RoboCup Soccer Simulator" (cf. figure. 2(e)) constitue la base standard et commune du déroulement des compétitions via Internet [CHE 03]. Le fait qu'il y ait une plate-forme unique de compétition par simulation a permis à ce simulateur de bénéficier d'un développement conséquent de la part des différentes équipes de recherche. Cependant, ce simulateur, malgré son aspect attrayant, reste à notre sens trop dédié aux compétitions de robots footballeurs, et les perceptions récupérées par les différents robots ne sont pas aussi modulables et réalistes que celles que nous souhaitons avoir dans nos simulations.

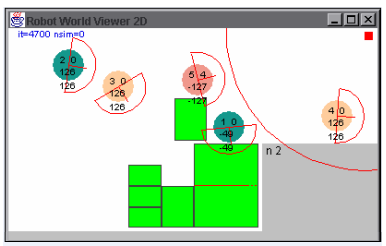
Le simulateur Webots [MIC 04] (cf. figure. 2(f)) commercialisé par la société CYBERBOTICS est à notre sens celui qui correspond le plus à nos besoins en simula-

4. Madkit "`http://www.madkit.org/`" est l'une des plates-formes de simulation SMA la plus aboutie.

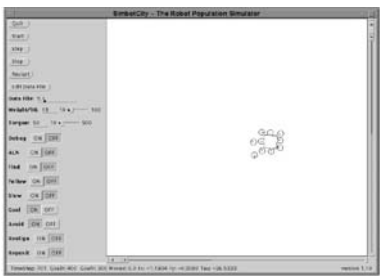


(a) Alia [DEL 93]

(b) Manta [DRO 93]



(c) Simulateur interfacé à Madkit [SIM 01]



(d) SimbotCity [KUB 97a]

(e) RoboCup Soccer Simulator [NOD 98]

(f) Webots [MIC 04]

Figure 2. *Illustrations de quelques simulateurs présentés dans la littérature*

tion. Il offre une large gamme de possibilités de programmation et d'interface. Il donne aussi à la fois : un affichage très réaliste, une interface d'édition du monde facile à utili-

ser, et prend en compte plusieurs types de contraintes physiques d'interaction entre les éléments des simulations. Sur un aspect multi-robots, cadre d'utilisation du simulateur qui nous intéresse plus particulièrement, Webots propose des processus de communication inter-robots locaux et globaux satisfaisants. Cependant, le déplacement des entités robotiques s'effectue d'une manière séquentielle, et utilise une base de temps fixe (caractéristiques observées pour les anciennes versions de Webots), c'est-à-dire que ceci implique que les entités virtuelles qui évoluent dans l'environnement n'ont pas les mêmes priorités de mouvement et par conséquent d'accès aux ressources (cf. section 6.3.1).

Il est à noter que la plupart des programmes sources des simulateurs présentés ci-dessus ne sont pas en libre accès, et que les documents qui les décrivent se restreignent généralement à un niveau de détail très faible. Par conséquent, il n'a pas été possible, ni d'utiliser l'un de ces simulateurs afin de l'adapter en fonction de nos exigences en simulation, ni même de nous inspirer efficacement de l'expérience de modélisation acquise par leurs concepteurs afin de concevoir le simulateur *MiRoCo*.

4. Le simulateur *MiRoCo*

Le fait qu'il n'y ait *a priori* aucun simulateur dans la littérature qui soit à la fois accessible et qui réponde au cahier des charges que nous nous sommes fixés (cf. section 6.3) nous a amenés à développer le simulateur *MiRoCo* (cf. figure. 3).

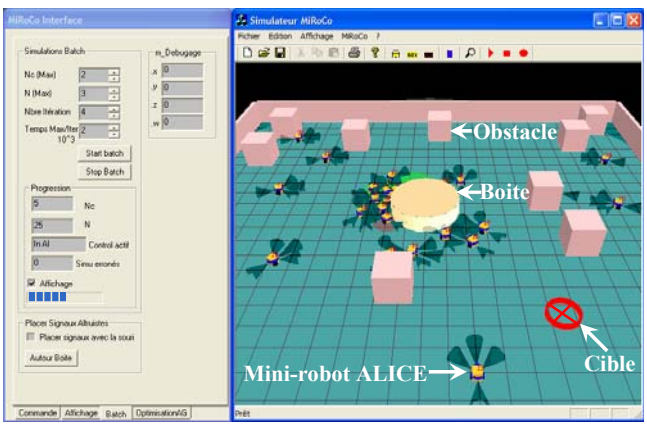


Figure 3. *Simulateur MiRoCo "MiniRobotique Collective"*

Parmi les points importants à respecter pour simuler correctement un système multi-agents situé, nous pouvons citer :

– la nécessité de séparation des agents de leur environnement. C'est ce qui est caractérisé dans [MAG 96] par une forme de dualité entité-environnement,

– la gestion efficace et réaliste de l'évolution temporelle des simulations, car une simulation n'est rien d'autre qu'une suite d'interactions consécutives entre agents. Dans le cadre des simulations que nous voulons effectuer, ceci concerne : le déplacement simultané des robots tout en respectant les contraintes de non chevauchement entre agents, l'émission-réception des signaux, le déplacement des robots par rapport aux déplacements des autres agents déplaçables, tels que les objets à pousser dans le cadre de la TCPO,

– la modélisation précise ou du moins réaliste des lois physiques intervenant lors de l'évolution et des interactions entre agents, car c'est ceci qui conditionne en grande partie, la correspondance entre la dynamique d'évolution du système multi-robots obtenue et celle qui règne dans le monde réel. Magnin dans [MAG 96] modélise les interactions entre agents par des règles environnementales (e.g., le rebond d'une balle sur les murs dans le cas des simulations de robots footballeurs). Cependant, ces règles peuvent être d'un nombre très important pour caractériser l'ensemble des interactions possibles, donc il convient de faire attention à leurs définitions pour ne pas risquer d'avoir des contradictions ou des aberrations par rapport au déroulement des simulations.

Dans ce qui va suivre, nous présentons les grandes lignes des principaux modèles implémentés dans *MiRoCo*, ainsi que l'organisation et les interactions existant entre les différentes classes constituant le simulateur.

5. Conception *orientée objet* de *MiRoCo*

Parmi les avantages majeurs d'une conception *orientée objet*, nous trouvons le fait qu'elle crée un lien quasi immédiat entre, d'une part les structures du domaine d'application et leurs mécanismes d'interaction[5] et d'autre part, la structure même du programme informatique (variables, structures, méthodes, etc.). Ceci induit par conséquent une démarche de transposition aisée de ce qui correspond au monde réel vers sa représentation informatique. Nous trouvons aussi une multitude d'autres avantages liés à la conception orientée objet dont bénéficie d'ailleurs énormément le domaine du génie logiciel. Nous pouvons citer parmi eux : la constructibilité ascendante des programmes ; la facilité de maintenance, d'évolution et de modification. *MiRoCo* est issu de cette conception *orientée objet* et il est dédié pour des simulations en robotique collective d'une manière générale.

6. Classes et organisation des simulations sous *MiRoCo*

La figure 4 représente l'ossature organisationnelle globale de *MiRoCo*, qui est décrite ici en utilisant une représentation UML (Unified Modelling Language) sous forme d'un diagramme de classes [BOO 00].

5. Dans notre cas, ce sont les structures et les mécanismes d'un système multi-robots.

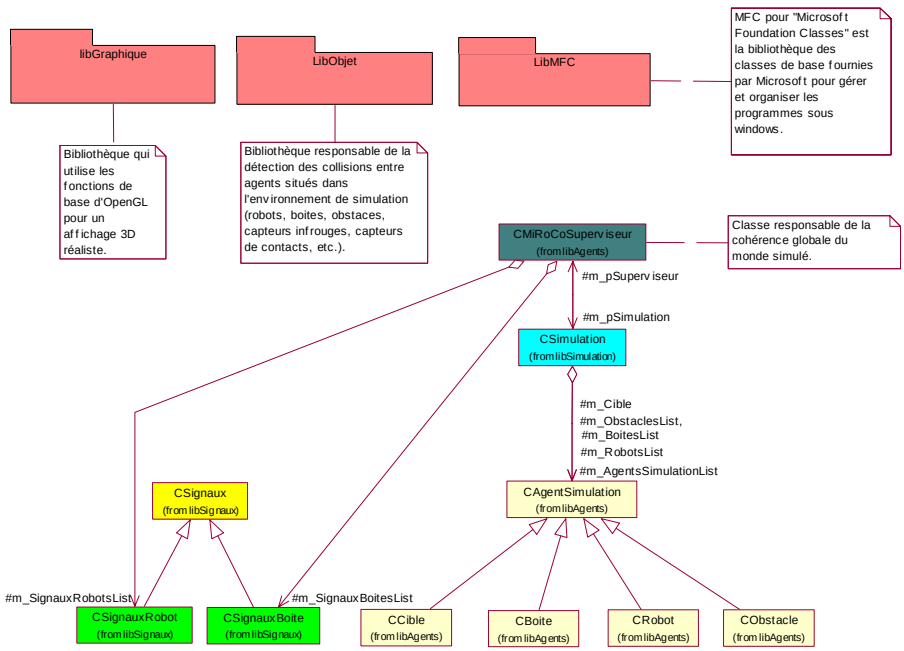


Figure 4. *Schéma UML représentant les principales classes intervenant dans le cadre des simulations sous* MiRoCo.

Nous décomposons les agents situés dans l'environnement en trois catégories :

– les dynamiques et/ou actifs (les robots) qui ont une complète autonomie de décision et d'action,

– ceux plutôt pseudo-statiques et/ou passifs (e.g., la boite) qui se déplacent uniquement sous l'influence d'autres agents, en l'occurrence sous l'action des robots,

– et enfin ceux qui sont statiques (les obstacles, les murs, la cible, etc.) qui ne changent pas de position ni d'orientation tout au long de la simulation.

6.1. *Agent robot*

La figure 5 correspond à la représentation logicielle d'un robot physiquement situé dans son environnement. L'agent robot est donc constitué de tous les éléments qui vont lui permettre de percevoir (les capteurs), analyser (le contrôleur) et enfin agir sur son environnement (en utilisant ses moteurs, et son émetteur de signaux).

Il est à noter que le robot mobile choisi dans la cadre de notre plate-forme expérimentale est le mini-robot ALICE [CAP 02] (cf. figure. 6(a)). C'est pour cette raison que nous avons tenu à ce que le robot simulé (cf. figure. 6(b)) corresponde le plus

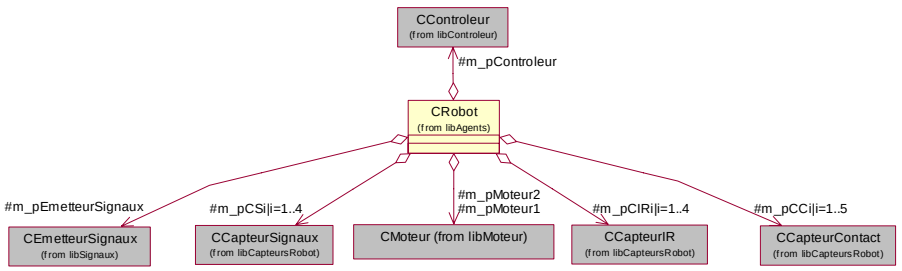
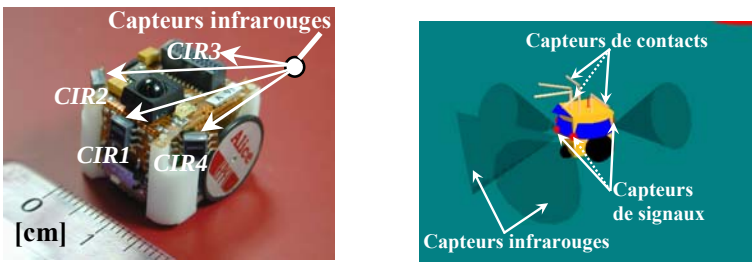


Figure 5. *Schéma UML de l'organisation des classes d'un mini-robot ALICE simulé*

fidèlement à la structure de base d'ALICE (dimension, forme conique et étendus des capteurs infrarouges), augmentée de certains capteurs nécessaires à la réalisation de la TCPO.



(a) ALICE (b) ALICE simulé

Figure 6. *Modélisation du mini-robot ALICE*

6.2. *Agents boite, obstacle, cible*

Les autres types d'agents situés dans l'environnement de simulation, correspondent aux :

– *agents boite* : contrairement à l'agent robot qui peut agir sur son environnement d'une manière autonome, l'agent boite ne peut se déplacer que sous l'impulsion des actions exercées par les autres agents robots. Cette agent dispose néanmoins d'un émetteur de signaux pour que les autres robots puissent le repérer dans l'environnement,

– *agents obstacle* : ces agents correspondent à des éléments statiques dans l'environnement, et sont censés ne pas être déplacés tout au long de la simulation,

– *agent cible* : correspond à un cercle avec une position et un rayon fixe dans l'environnement. Cet agent particulier permet par exemple, dans le cas de la TCPO, de connaître la zone où doit être acheminé l'objet à pousser. Nous considérons qu'une

simulation est achevée à partir du moment où l'objet à pousser touche le pourtour du cercle représentant la cible.

6.3. *Classe superviseur*

Cette classe joue un rôle primordial lors des simulations. En effet, c'est elle qui dicte les lois du monde simulé. Ces lois sont définies de façon à tendre vers les lois physiques qui règnent dans le monde réel. C'est donc cette classe qui fait en sorte que les simulations (mouvement des agents, interactions entre agents, gestion des signaux dans l'environnement, etc.) soient les plus réalistes et cohérentes possibles. Notons que cette classe pourrait jouer aussi le rôle d'un contrôleur central dans le cas où l'on envisagerait de procéder à ce type de contrôle.

Dans ce qui suit les principaux mécanismes et modèles implémentés dans la classe superviseur sont exposés.

6.3.1. *Moteur physique*

Pour simuler un environnement multi-robots où les déplacements des robots mènent bien souvent à des conflits spatiaux[6], il est primordial de pouvoir respecter un certain nombre de contraintes afin d'essayer de faire tendre l'exécution des simulations vers ce qui ce passe effectivement dans le monde réel. Ces contraintes sont illustrées dans ce qui suit via un certain nombre d'exemples et de travaux.

– *Contrainte de non-séquentialité des actions des agents* : pour que les agents ne se rentrent pas les uns dans les autres, la solution la plus immédiate consiste à exécuter les actions des agents dans un ordre immuable (séquentiel) et à vérifier à chaque fois que l'agent déplacé ne rentre pas dans les autres agents. Cependant, ceci introduit une relation de précédence indésirable, puisque l'action du robot n-1 est toujours exécutée avant celle du robot n. Ainsi, si deux agents ou plus sont régulièrement en compétition pour une ressource (e.g., volume de l'objet à pousser), alors celui ayant son action au rang n-1 prendra systématiquement la ressource aux agents de rangs supérieurs. D'où l'idée inhérente à cette contrainte qui impose que les actions des robots doivent s'exécuter simultanément (en parallèle) afin qu'il n'y ait aucune relation de hiérarchie d'accès aux ressources si toutefois elles sont sollicitées en même temps,

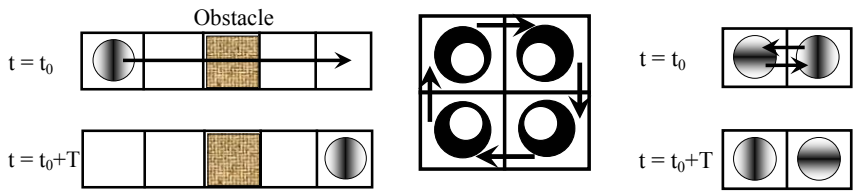
– *Contrainte de non chevauchement des agents entre eux* : cette contrainte est importante dans le sens où les éléments du monde physique ne peuvent pas se chevaucher. Par conséquent, ce qui est exigé par cette contrainte est le fait que les agents doivent toujours avoir leurs volumes (et/ou surfaces) complètement disjoints.

L'un des moyens couramment utilisés en simulation d'agents autonomes situés est de décomposer l'environnement d'évolution des agents en un certain nombre de positions discrètes [DRO 93], [DEL 93]. Ainsi, à chaque pas d'échantillonnage, chaque

6. On appelle conflit spatial, une situation où plusieurs agents tentent d'utiliser simultanément un même espace alors que celui-ci est insuffisant pour le permettre.

agent occupera une case bien précise de l'environnement qui sera bien évidemment différente de celles des autres agents qui partagent le même environnement. Cependant, même en discrétisant l'environnement, l'évolution des agents n'est pas à l'abri de certaines situations conflictuelles qui doivent être gérées.

La figure 7(a) par exemple montre le cas où la position finale d'un agent est plausible, cependant comme l'agent traverse au préalable un obstacle, alors la position finale de l'agent devient complètement aberrante et l'algorithme implémenté doit détecter cela. Dans le cas de la figure 7(b), chaque agent souhaite se déplacer dans la case de son voisin, ce qui semble en théorie possible. Néanmoins, lors de la simulation, quel que soit l'agent déplacé en premier, il apparaîtra que sa case voisine est déjà occupée, sauf si l'on prend pour principe d'exécuter toutes les actions en parallèle, avec éventuellement un "retour en arrière" en cas d'incohérence finale. Cet algorithme peut présenter néanmoins des résultats non satisfaisants, par exemple lorsque deux agents souhaitent permuter leurs positions (cf. figure. 7(c)), on aura un résultat final cohérent, cependant le croisement des deux agents devrait être impossible, mais ceci n'est pas détecté.



(a) Agent se déplaçant de quatre cases à travers un obstacle !

(b) Échanges de positions non conflictuelles.

(c) Échange de positions conflictuelles !

Figure 7. *Quelques situations conflictuelles induites par les déplacements des agents [MAG 96].*

Il serait tout à fait possible de présenter d'autres exemples posant problème. Cependant, l'objectif ici n'est pas de faire une présentation exhaustive des différents conflits spatiaux qui peuvent survenir lors de l'utilisation d'un environnement discret, mais est de montrer, via des exemples assez simplistes, des cas de conflits spatiaux qui doivent être résolus afin d'aboutir à des simulations cohérentes et précises.

Par conséquent, afin d'aboutir à la cohérence des mouvements des agents présents dans les simulations, il faut que ces agents, d'une part, ne passent pas les uns au travers des autres, et d'autre part, il faut que leurs actions puissent s'exécuter en même temps simultanément et non pas séquentiellement. Cette deuxième contrainte est très importante dans le cas de la TCPO, car nous avons plusieurs agents qui partagent un milieu très confiné (en l'occurrence le pourtour immédiat de l'objet à pousser). Nous notons que dans le cas où l'on aurait donné une forme de séquentialité (donc de priorité) au mouvement des agents, alors cela conduirait inévitablement à une dynamique d'évolution du système multi-robots tout à fait faussée.

Avant d'expliquer les détails du *moteur physique* que nous avons implémenté sur *MiRoCo*, nous donnons au préalable le modèle de déplacement des agents dans l'environnement continu[7] : à l'instant d'échantillonnage $t + \triangle t$, chaque agent mobile va évoluer de son état initial ($Position_t$, α_t) $\equiv$ (x_t, y_t, α_t) vers un état final ($Position_{t+\triangle t}$, $\alpha_{t+\triangle t}$) $\equiv$ ($x_{t+\triangle t}$, $y_{t+\triangle t}$, $\alpha_{t+\triangle t}$) (cf. figure. 8) suivant cette équation :

$$\begin{cases} Position_{t+\Delta t} = Position_t + \vec{V_t} \times \Delta t \\ \alpha_{t+\triangle t} = \alpha_t + W_t \times \Delta t \end{cases} \quad (1)$$

Avec :

– $\vec{V_t}$ et W_t correspondant respectivement à la vitesse linéaire et angulaire instantanée de l'agent.

– α_t l'angle que fait l'agent avec la verticale au repère absolu lié à l'environnement.

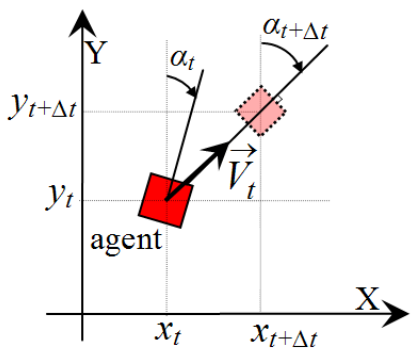


Figure 8. *Modèle de déplacement des agents mobiles dans les simulations*

L'algorithme 1 résume en pseudo-code le *moteur physique* implémenté dans *MiRoCo* qui respecte les deux contraintes mentionnées ci-dessus. La figure 9 nous donne une représentation graphique du fonctionnement de cet algorithme pour un cas simple d'un conflit spatial entre deux agents mobiles.

Comme cela est indiqué dans l'algorithme 1, la précision de l'algorithme augmente proportionnellement avec la valeur de la constante *Précision*. Cette constante permet de décomposer les trajectoires futures (effectuées en un pas d'échantillonnage) des agents mobiles en *Précision* (le nombre) parties égales (cf. figure. 9(c)). Une partie est calculée par exemple pour le cas de l'agent$_i$ par Pas_i = [(1$\diagup$*Précision*) $\times$ $Vitesse_i$ $\times$ Δt]. L'idée consiste ensuite à trouver de combien de Pas_i faire reculer la position

7. C'est-à-dire qu'un agent peut occuper n'importe quelle position (x, y) de l'environnement.

Algorithme 1 *Moteur physique* implémenté dans *MiRoCo*

Entrée : int *Précision* = const1 ; //Donne la précision de l'algorithme. Plus const1 est grande plus la précision de l'algorithme est plus importante.//

Pour chaque agent mobile dans l'environnement **Faire**
$PositionInitiale_i = PositionAgent_i$; //Sauvegarder sa position initiale
$PositionFinale_i = PositionInitiale_i + Vitesse_i \times \Delta t$; //Calcul de sa position finale. Avec $Vitesse_i$ correspondant à la vitesse de l'agent$_i$.//
$PositionAgent_i = PositionFinale_i$; //Affectation de la position finale à l'agent en question
$ReculeMax_i = 0$; //$ReculeMax_i$ correspond au nombre d'incréments en moins à faire par l'agent$_i$ par rapport à sa position finale pour ne pas être en collision avec d'autres agents.//
Fin Pour

//Obtention de $ReculeMax_i$ effective pour chaque agent$_i$.//
Pour tous les agents présents dans l'environnement **Faire**
//NB : les indices $_i$ et $_j$ désignent deux agents distincts.
int $Recule = 0$;
Tant que agent$_i$ est en collision avec l'agent$_j$ **Faire**
$Recule = Recule + 1$;
$PositionAgent_i = PositionFinale_i$ - [($Recule$ / *Précision*) $\times\ Vitesse_i \times \Delta t$] ;
$PositionAgent_j = PositionFinale_j$ - [($Recule$ / *Précision*) $\times\ Vitesse_j \times \Delta t$] ;
Fin Tant que

Si $Recule > ReculeMax_i$ **Alors**
$ReculeMax_i = Recule$;
Fin Si

Si $Recule > ReculeMax_j$ **Alors**
$ReculeMax_j = Recule$;
Fin Si

//Réaffection des positions finales des agents afin de faire d'autres tests de collision avec d'autres agents.
$PositionAgent_i = PositionFinale_i$;
$PositionAgent_j = PositionFinale_j$;
Fin Pour

//Obtention des positions finales des agents mobiles qui respectent, d'une part, l'évolution parallèle (simultanée) des robots et d'autre part, la contrainte de non chevauchement des agents entre eux.//

Pour tous les agents mobiles dans l'environnement **Faire**
$PositionAgent_i = PositionFinale_i$ - [($ReculeMax_i$ / *Précision*) $\times\ Vitesse_i \times \Delta t$] ;
Fin Pour

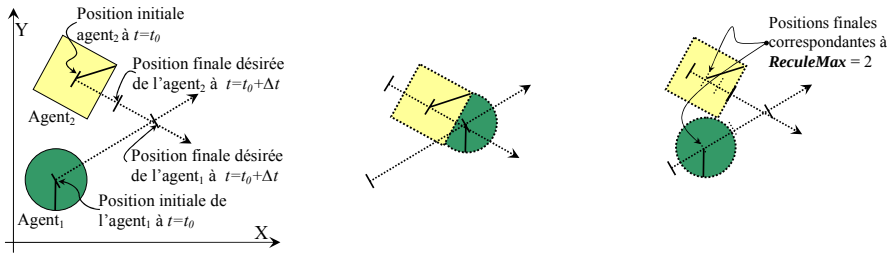
---

future de l'agent$_i$ pour qu'il ne soit pas en collision avec aucun autre agent situé[8] (ce qui correspond dans l'algorithme 1 à trouver $ReculeMax_i$ de l'agent$_i$). Nous notons que si $ReculeMax_i$ = *Précision*, alors le robot revient à sa position initiale, et si $ReculeMax_i = 0$, alors ceci implique que la position future du robot$_i$ se fait sans aucune collision.

6.3.2. *Modèle de déplacement de l'objet à pousser*

Les déplacements de l'objet à pousser dans le cas de la TCPO s'effectuent entièrement sous l'impulsion des actions exercées par les agents robots. Nous avons modélisé le déplacement de l'objet à pousser de telle sorte que le modèle tienne compte des ac-

8. Ceci peut se faire aussi en utilisant des méthodes analytiques telle que la dichotomie pour accélérer cette recherche.



(a) Positions initiales et finales désirées (b) Mouvement sans l'application du *moteur physique* (c) Mouvement avec l'application du *moteur physique* (*Précision*=3)

Figure 9. *Application du* moteur physique *implémenté dans* MiRoCo *pour un cas simple de deux agents en mouvement.*

tions simultanées de plusieurs robots à la fois (robots qui sont en interaction directe avec l'objet en question à l'instant t). Le modèle est résumé dans ce qui suit :

Le déplacement de l'objet à pousser entre les instants d'échantillonnage t et $t+\Delta t$ est conditionné, d'une part, par le nombre de robots en collision avec la boite "*Nrcb*" à l'instant t et "*Nc*" (nombre minimum de robots pour pouvoir déplacer l'objet), et d'autre part, par la résultante de vitesse $\overrightarrow{V_b}$ engendrée par ces robots à l'instant t (cf. équation. 2). Ces deux contraintes se résument ainsi :

Si (*Nrcb* < *Nc*) **Alors** la boite à pousser reste immobile.

Sinon la boite se déplace en fonction de $\overrightarrow{V_b}$ (cf. figure. 10(a)), correspondant à l'application du principe de conservation de la *quantité de mouvement* avant et après impact pour le cas de collisions élastiques.

$$\overrightarrow{V_b} = \frac{Nrcb \times m_r}{M_b} \times \sum_{i=1}^{Nrcb} \overrightarrow{V}_{ri} \tag{2}$$

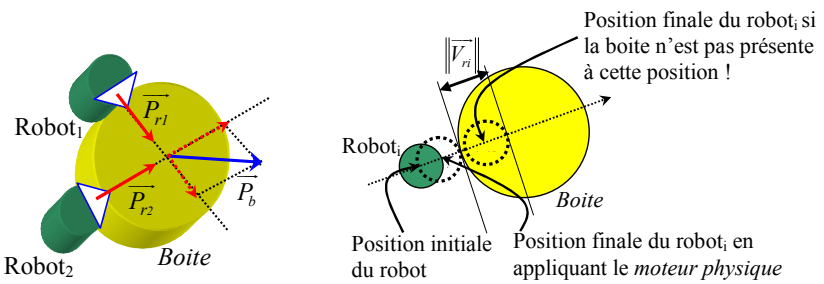
Avec :

– m_r masse d'un robot élémentaire,

– M_b masse de la boite à pousser,

– $\overrightarrow{V}_{ri}$ vitesse du robot$_i$ quand il rentre en collision avec la boite à pousser, telle que : $0 < \left\| \overrightarrow{V}_{ri} \right\| \leq (Vitesse_i \times \Delta t)$ (cf. figure. 10(b)).

Ainsi, à chaque pas d'échantillonnage, si le nombre de robots en collision avec la boite dépasse *Nc*, alors celle-ci va se déplacer à la vitesse instantanée $\overrightarrow{V_b}$, déterminée



(a) Conservation de la quantité de mouvement.

(b) Obtention du module de la vitesse du robot au moment de l'impact avec la boite.

Figure 10. *Modèle de déplacement de la boite*

par les multiples collisions des robots avec elle. Dans les simulations effectuées sous *MiRoCo*, ce déplacement peut être entravé s'il y a au moins un agent qui gêne sa progression.

6.3.3. *Modèle d'échange de signaux entre agents*

Un autre élément important modélisé sous *MiRoCo* concerne les différents signaux émis et/ou reçus par les agents présents dans l'environnement, en l'occurrence les robots, le ou les objet(s) à pousser et la cible à atteindre. Effectivement, l'objet à pousser, comme la cible à atteindre, dispose de balises émettrices caractéristiques qui permettent aux robots de les repérer et de les distinguer dans l'environnement (cf. figure. 1). Les signaux altruistes constituent l'autre type de signaux utilisés par les robots pour une communication de très bas niveau entre eux [ADO 04b]. Ces signaux altruistes doivent être modélisés de la manière la plus réaliste possible afin de permettre une analyse fiable de leur pertinence pour la réalisation des tâches coopératives entreprises.

Les points communs entre les différents types de signaux émis par les agents dans l'environnement sont :

– un champ d'émission (portée) déterminé en fonction de l'agent et/ou de sa situation dans l'environnement,

– une intensité décroissante à chaque fois qu'on s'éloigne de leur centre émetteur.

Dans ce qui va suivre, nous présentons les deux modélisations de signaux implémentées sous *MiRoCo*. Il est à noter que ces deux types de modélisation utilisent une structure qui fait office de tableau noir "*blackboard*" [HAY 84], [ENG 88] afin de centraliser et de sauvegarder temporairement les informations utiles des signaux. Les robots n'ont alors qu'à émettre une requête vers cette structure pour pouvoir lire les informations locales qui les concernent.

– **Signaux à décroissance d'intensité discrète :** Dans cette première modélisation, les signaux émis dans l'environnement sont représentés de manière discrète. C'est-à-dire que nous avons décomposé l'environnement en ($N_{li} \times N_{co}$) cases égales, où chaque case correspond à un élément du tableau TAB$_s$ de N_{li} lignes et N_{co} colonnes. Chaque case du tableau contient autant de valeurs d'intensité de signaux qu'il y a de types d'agents qui émettent des signaux couvrant cette case du tableau [9]. Il est à noter qu'un signal sauvegardé dans la case du tableau garde une trace sur le type d'agent (robot, boite ou cible) qui a émis le signal. Nous notons aussi que l'intensité du signal est maximale dans la case du tableau contenant la position du centre de l'agent émetteur et elle diminue avec un facteur γ à chaque fois qu'on s'éloigne d'une case (dans toutes les directions) par rapport au centre de l'agent (cf. figure. 11).

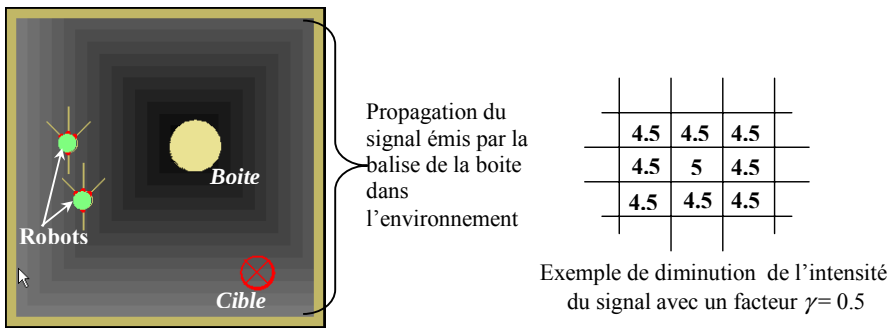


Figure 11. *Émission de signaux à valeurs discrètes dans l'environnement*

Ainsi, pour faire fonctionner le capteur de signaux "CS_i" (appartenant à un robot déterminé de la simulation), il n'a qu'à lire à la position (x_i, y_i) qu'il occupe dans l'environnement, la valeur du signal pour lequel il est dédié.

Cette version d'émission de signaux, qui utilise l'environnement comme médium de communication, est fortement inspirée des phéromones émises par les fourmis. Ces phéromones leur permettent entre autres de garder une trace du trajet à faire pour atteindre par exemple une source de nourriture. Cette trace est amplifiée par le passage répété des fourmis, qui sont de plus en plus nombreuses à emprunter ce chemin.

Inconvénients : Cette première approche de modélisation des échanges de signaux entre agents s'avère être très coûteuse en temps de calcul et en mémoire. Ceci est accentué de plus par le fait que nous voulons disposer d'informations très précises sur l'évolution des intensités des signaux à chaque fois qu'on s'éloigne de leur centre émetteur. En effet, ceci impose l'utilisation d'un tableau TAB$_s$ de très grande dimension. L'autre point négatif de cette modélisation est la lourdeur de rafraîchissement des cases de TAB$_s$ à chaque pas d'échantillonnage. Ceci est dû principalement aux

9. La case en question contiendra s'il y a plusieurs signaux de même nature (i.e., émis par des agents identiques), la valeur de la somme des intensités des signaux émis par ces agents.

déplacements des agents émetteurs de signaux et à la disparition de ces signaux dans l'environnement (car ils sont censés être volatiles).

L'implantation de ce type d'approche d'émission/réception de signaux, en utilisant l'environnement comme médium de communication, est aussi techniquement difficile à maîtriser en robotique mobile en environnement réel, car ceci suppose de manipuler des substances chimiques, donc nécessite des capteurs et des récepteurs dédiés.

Tous les inconvénients cités ci-dessus nous ont amenés à proposer une autre modélisation pour la gestion de l'émission/réception des signaux dans l'environnement.

– **Signaux à décroissance d'intensité continue :**

Dans cette seconde modélisation, nous avons opté pour une gestion des signaux émis et/ou perçus par les différents agents dans l'environnement, nettement moins coûteuse en temps de calcul et en espace mémoire.

Nous établissons pour cela une liste de signaux présents dans l'environnement, avec pour chaque signal les caractéristiques décrites dans l'algorithme 2. Une représentation graphique de ces signaux dans l'environnement est donnée en figure 12.

Algorithme 2 Méthode d'émission de signaux continus

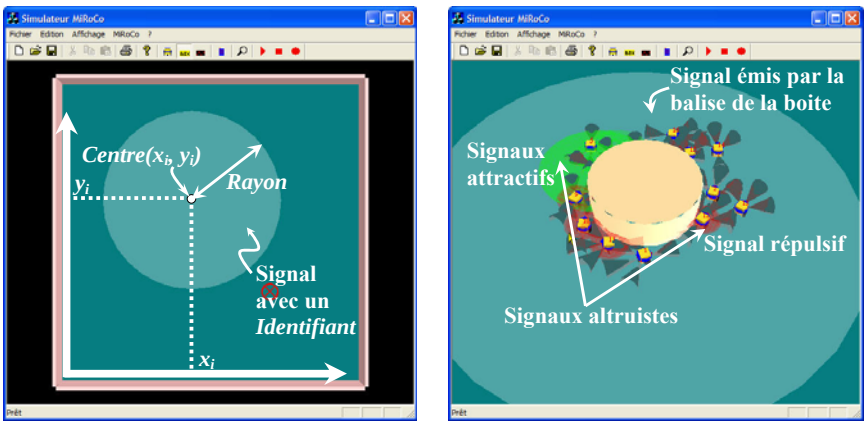
Entrée : Disposer l'agent$_i$ d'un émetteur de signaux ;
Si l'agent$_i$ doit émettre un signal **Alors**
Ajouter à la liste des signaux présents dans l'environnement un signal caractérisé comme suit :
- un centre "$Centre(x_i, y_i)$" correspondant à la position de l'agent$_i$ (cf. figure. 12(a)) ;
- un rayon "$Rayon$" déterminé par l'agent$_i$ (cf. figure. 12(a)) ;
- une intensité maximale "$IntensiteSignalCentre$" qui correspondra à l'intensité perçue au centre du signal ;
- une valeur de décroissance de la valeur absolue de l'intensité du signal "$DecroisanceSignal$", et ce quand on s'éloigne de son centre ;
- un identifiant "$Identifiant$" correspondant à celui de l'agent$_i$;
Fin Si

Pour ce qui est de l'algorithme 3, il décrit le mécanisme utilisé par les capteurs de signaux, afin de récupérer les bonnes valeurs d'intensité des signaux émis dans l'environnement.

NB : La liste des signaux présents dans l'environnement est rafraîchie à chaque pas d'échantillonnage.

6.4. *Classe Simulation*

Cette classe contient les listes correspondant aux différents types d'agents présents dans l'environnement, en l'occurrence la liste des robots, des boites et des obstacles. Cette classe contient aussi une instance de la classe superviseur, et un agent cible. La largeur et la longueur de l'environnement simulé, la hauteur et la largeur des murs sont aussi définies dans cette classe. Il faut donc au moins une instance de cette classe pour disposer de tous les éléments nécessaires à l'exécution d'une simulation sous *MiRoCo* (cf. figure. 4).



(a) Caractéristique d'un signal émis dans l'environnement.

(b) Émission de signaux de différents types dans l'environnement.

Figure 12. *Signaux à décroissance continue*

Algorithme 3 Méthode de réception des signaux continus

Entrée : Disposer l'agent$_i$ d'un ou plusieurs récepteurs de signaux ;
Pour tous les récepteurs de l'agent$_i$ **Faire**
  Pour tous les signaux présents dans l'environnement **Faire**
    Si $(PositionRcpteur$ - $CentreSignal) \leq RayonSignal$ **Alors**
      Si $IntensiteSignalCentre > 0$ **Alors**
        //Dans le cas d'un signal altruiste attractif ou bien d'un signal émanant de la boite et ou de la cible
        $ValeurRecepteur$=$IntensiteSignalCentre$ - $DecroisanceSignal \times DistanceDuCentreDuSignal$;
      Sinon
        //Sinon c'est un signal répulsif
        $ValeurRecepteur$ = $IntensiteSignalCentre$ + $DecroisanceSignal \times DistanceDuCentreDuSignal$;
      Fin Si
    Fin Si
  Fin Pour
Fin Pour

6.5. *Bibliothèques externes utilisées*

6.5.1. *Environnement de programmation*

Nous avons développé *MiRoCo* en utilisant Visual C++ 6.0. Cette plate-forme de programmation pour C++ propose une architecture de programme, appelée document/vue basée sur les MFC "*Microsoft Foundation Class Library*", qui consiste à séparer les données (document) de leur visualisation (vue). L'idée générale a été alors d'intégrer les classes modélisant le système multi-robots, à l'ossature standard d'une architecture document/vue et ce afin de bénéficier des fonctionnalités des MFC [HOR 99].

6.5.2. *Affichage*

L'aspect affichage graphique de l'environnement multi-robots de *MiRoCo* est complètement géré en utilisant la bibliothèque graphique OpenGL "*Open Graphics Library*", ce qui nous a permis de bénéficier de ses différentes potentialités. Parmi elles, nous pouvons citer, le choix des angles de vues "*ViewPoint*" et de leur nombre pour visualiser la scène. Ainsi, nous pouvons par exemple équiper chaque robot d'une caméra individuelle, qui lui permettra (via un traitement d'images approprié) de retirer des informations pertinentes sur son environnement (cf. figure. 13).

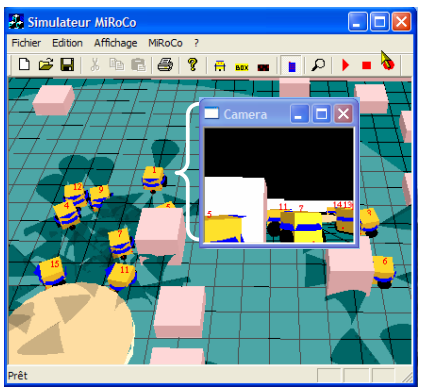


Figure 13. *Angle de vue à partir de la position du robot numéro 1*

La scène d'évolution des robots peut être composée de plusieurs volumes élémentaires (cube, sphère, etc.), ou de formes complexes (qui peuvent être construites par accumulation de formes de base). Ces formes pouvant être colorées, texturées, avoir une apparence d'un matériau quelconque, etc. OpenGL offre donc un grand nombre de fonctionnalités d'affichage, rendant ainsi le rendu des scènes (contenant tous les éléments participant aux simulations : robots, boites, obstacles, signaux émis par les agents, etc.) à observer très réaliste. Ceci nous permet par conséquent d'avoir des indications très précises sur la dynamique d'interaction et d'évolution du système multi-robots simulé.

6.5.3. *Détection de collisions*

Il est nécessaire de disposer d'outils qui détectent automatiquement les collisions éventuelles entre les différents agents situés dans l'environnement. Ce besoin nous a conduit à rechercher une bibliothèque appropriée de détection des collisions [REG 02]. Nous avons opté finalement pour la bibliothèque SOLID[10] [BER 04] qui est diffusée sous une licence "*GNU Library General Public Licence*". Les différents agents qui peuvent être testés grâce à SOLID, peuvent avoir des formes élémentaires diverses telles que des cubes, sphères, cylindres, cônes, etc. Mais ils peuvent aussi être

10. `http://www.win.tue.nl/~gino/solid/`

construits par une adjonction de formes élémentaires afin de produire des agents aux formes complexes, comme le cas d'un robot par exemple.

7. Déroulement d'une simulation complète pour le cas de la *TCPO*

Le déroulement des simulations sous *MiRoCo* pour la cas de la TCPO est résumé par les étapes décrites dans l'algorithme 4.

Algorithme 4 Étapes nécessaires pour l'exécution de la TCPO sous *MiRoCo*

Début de la simulation multi-robots : Donner une configuration initiale à tous les agents présents dans l'environnement ;
Tant que la simulation n'est par terminée **Faire**
- Rafraîchissement les signaux présents dans l'environnement ;
- Déplacement de chaque robot dans l'environnement en fonction, d'une part des stimuli locaux qu'il perçoit et d'autre part de l'architecture de contrôle qu'il utilise ;
- Application du *moteur physique* pour avoir des déplacements cohérents et physiquement réalisables ;
- Déplacement éventuel de la boite si les conditions nécessaires à son déplacement sont vérifiées ;
- Affichage de la scène avec les différents agents ;

Fin Tant que

8. Conclusion et perspectives

La conception modulaire du simulateur *MiRoCo*, nous a permis de maîtriser la complexité inhérente aux simulations de systèmes multi-robots. Les différentes modélisations implémentées dans *MiRoCo* nous ont permis d'obtenir une précision importante par rapport à l'évolution de la dynamique du système multi-robots. *MiRoCo* constitue à présent un outil complet pour simuler des systèmes multi-robots.

Dans le cadre des simulations effectuées dans [ADO 04a], [ADO 04b] et [ADO 05b], *MiRoCo* s'est avéré un outil précieux pour étudier et quantifier d'une part, les caractéristiques des architectures de contrôle implémentées sur les robots et d'autre part, les phénomènes et comportements émergents induits par le coopération des robots.

Plusieurs améliorations sont envisageables pour rendre les simulations sous *MiRoCo* d'autant plus rapides, plus précises et plus conviviales qu'elles le sont actuellement. Parmi les pistes envisageables, citons le fait d'optimiser davantage les programmes du *moteur physique* afin d'avoir des temps de calculs moins importants, ou bien l'identification plus précise des caractéristiques structurelles du robot utilisé (capteurs, actionneurs, communication, etc.) pour améliorer la modélisation faite du robot dans *MiRoCo*. Nous pourrons ainsi obtenir des simulations plus précises des dynamiques des système multi-robots. Il serait aussi envisageable d'intégrer la librairie ODE "*Open Dynamique Engines*" à *MiRoCo* afin de lui permettre une utilisation aisée de plusieurs modélisations liées à la physique d'interaction entre agents (frottement, amortissement, etc.).

Du point de vue de la convivialité d'utilisation de *MiRoCo*, là aussi nous pouvons envisager plusieurs améliorations qui peuvent se résumer comme suit :

– l'ajout d'une interface utilisateur, qui permettra d'agencer et de modifier les architectures de contrôle proposées d'une manière plus conviviale, c'est-à-dire sans recourir à chaque fois à une modification des lignes de code. L'idée est d'utiliser des blocs de comportements pré-programmés et des liens appropriés pour construire intuitivement (e.g., à l'aide de la souris) l'architecture de contrôle à implémenter sur les robots,

– la génération automatique du code machine à implémenter sur les véritables robots, et ce à partir du contrôle implémenté et testé au préalable en simulation. Ceci dans le cadre du mini-robot ALICE correspondrait à générer le code machine du microcontrôleur PIC16F877.

9. Bibliographie

[ADO 04a] ADOUANE L., LEFORT-PIAT N., « Emergence of Group Intelligence from Minimalist Control of Mobile Mini-Robots », *35th International Symposium On Robotics ISR*, Paris-France, 23-26 March 2004, In CD.

[ADO 04b] ADOUANE L., LEFORT-PIAT N., « Hybrid Behavioral Control Architecture for the Cooperation of Minimalist Mobile Robots », *International Conference On Robotics And Automation*, New Orleans-USA, April 26 - May 1 2004, p. 3735-3740.

[ADO 05a] ADOUANE L., « Architectures de contrôle comportementales et réactives pour la coopération d'un groupe de robots mobiles », PhD thesis, Université de Franche-Comté, Laboratoire d'Automatique de Besançon UMR CNRS 6596, 11 avril 2005, Thèse N°1071.

[ADO 05b] ADOUANE L., LEFORT-PIAT N., « Methodolgy of Parameters Optimization for an Hybrid Architecture of Control », *16th IFAC World Congress*, Prague-Czech Republic, July 5-8 2005, In CD.

[ARK 92] ARKIN R. C., « Cooperation without Communication : Multi-agent Schema Based Robot Navigation », *Journal of Robotic Systems*, vol. 9, n° 3, 1992, p. pp.351-364.

[BAL 95a] BALCH T., ARKIN R. C., « Motor schema-based formation control for multiagent robot teams », LESSER V., GASSER L., Eds., *Proceedings of the First International Conference on Multiagent Systems (ICMAS'95)*, San Francisco, CA, USA, 1995, AAAI Press, p. 10–16.

[BAL 95b] BALCH T., ARKIN R. C., « Communication in reactive multiagent robotic systems », *Autonomous Robots*, vol. 1, n° 1, 1995, p. pp.27-52.

[BAL 03] BALDASSARRE G., NOLFI S., PARISI D., « Evolution of collective behaviour in a team of physically linked robots », *In R. Gunther, A. Guillot, and J.-A. Meyer, editors, Applications of Evolutionary Computing, Springer Verlag, Heidelberg, Germany,,* , 2003, p. 581-592.

[BER 04] BERGEN G. V. D., *Collision Detection in Interactive 3D Environments*, Elsevier, 2004.

[BOO 00] BOOCH G., RUMBAUGH J., JACOBSON I., *Le guide de l'utilisateur UML*, Eyrolles, 2000.

[BRO 90] BROOKS R. A., « Elephant Don't Play Chess », *Robotics and Automation Systems*, vol. 6, 1990, p. pp.3-15.

[CAP 02] CAPRARI G., ESTIER T., SIEGWART R., « Fascination of Down Scaling - Alice the Sugar Cube Robot », *Journal of Micro-Mechatronics*, vol. VSP-Utrecht, 2002, p. pp.177-189.

[CHE 94] CHEN Q., LUH J. Y. S., « Coordination and Control of a Group of Small Mobile Robots », *International Conference On Robotics And Automation*, 1994, p. 2315-2320.

[CHE 03] CHEN M., DORER K., FOROUGHI E., HEINTZ F., HUANG Z., KAPETANAKIS S., KOSTIADIS K., KUMMENEJE J., MURRAY J., NODA I., OBST O., RILEY P., STEFFENS T., WANG Y., YIN X., « RoboCup Soccer Server, 2003. User Manual for Soccer Server Version 7.07 and later », rapport, 2003.

[DEL 93] DELAYE C., « Structures et organisations des systèmes multi-agents autonomes et adaptatifs », PhD thesis, Université Paris VI, 1993.

[DRO 93] DROGOUL A., « De La Simulation Multi-Agent A La Résolution Collective de Problèmes – Une Étude De l'Émergence De Structures D'Organisation Dans Les Systèmes Multi-Agents », PhD thesis, Université Paris VI, Novembre 1993.

[ENG 88] ENGLEMORE E., MORGAN T., *Blackboard sytems*, adisson-Wesley, 1988.

[GUT 00] GUTKNECHT O., FERBER J., MICHEL F., « Madkit : une expérience d'architecture de plate-forme multi-agent générique », LA RÉUNION H., Ed., *8ème Journées Francophones sur l'Intelligence Artificielle Distribuée et les Systèmes Multi-Agents (JFIAD-SMA'2000)*, 2000, p. 223-236.

[HAY 84] HAYES-ROTH B., *A BlackBoard Model of Control. Technical Report*, Standford University, 1984.

[HOR 99] HORTON I., *Visual C++ 6*, Eyrolles, 1999.

[KUB 94] KUBE C., ZHANG H., « Stagnation Recovery Behaviours for Collective Robotics », *International Conference on Intelligent Robots and Systems*, 1994, p. 1893-1890.

[KUB 97a] KUBE C., « Collective Robotics : From Local Perception to Global Action », PhD thesis, Univeristy of Alberta Canada, 1997.

[KUB 97b] KUBE C., ZHANG H., « Task Modelling in Collective Robotics », *Autonomous Robots*, vol. 4, n° 1, 1997, p. pp.53-72.

[KUB 00] KUBE C., BONABEAU E., « Cooperative transport by ants and robots », *Robotics and Autonomous Systems*, vol. 30, n° 1/2, 2000, p. pp.85-101, Elsevier.

[MAG 96] MAGNIN L., « Modélisation et simulation de l'environnement dans les systèmes multi-agents », PhD thesis, Université Paris VI, novembre 1996.

[MAT 94] MATARIC´ M. J., « Interaction and Intelligent Behavior », PhD thesis, MIT, 1994.

[MEL 01] MELHUISH C., *Strategies for Collective Minimalist Mobile Robots*, vol. ERS 6, Professsional Engineering Publishing, 2001.

[MIC 04] MICHEL O., « Cyberbotics Ltd - WebotsTM : Professional Mobile Robot Simulation », *International Journal of Advanced Robotic Systems*, vol. 1, n° 1, 2004, p. pp.39-42.

[MüL 98] MÜLLER J., PETIN J., MOREL G., VACHON B., PEGARD C., BRASSART E., HUTIN N., DEMBELÉ S., JANEX A., MORELO B., RAVASSARD J., BOURJAULT A., « Conception de systèmes de transport collectif d'objets », *Premières Journées du Pôle Microrobotique*, 1998, p. 61-66.

[MUñ 03] MUÑOZ A., « Coopération située : une approche constructiviste de la conception de colonies de robots », PhD thesis, Université Pierre et Marie Curie, Paris VI, April 2003.

[NOD 98] NODA I., MATSUBARA H., HIRAKI K., FRANK I., « Soccer Server : a tool for research on multi-agent systems », *Applied Artificial Intelligence*, 1998.

[PAR 01] PARKER L. E., « Evaluating Success in Autonomous Multi-Robot Teams : Experiences from ALLIANCE Architecture Implementations », *Journal of Theoretical and Experimental Artificial Intelligence*, vol. 13, 2001, p. pp.95-98.

[PRE 90] PREMVUTI S., YUTA S., « Consideration on the Cooperation of Multiple Autonomous Mobile Robots », *International Workshop on Intelligent Robots and Systems*, Ikabari-Japan, July 1990, p. 59-63.

[RAM 94] RAM A., ARKIN R., BOONE G., PEARCE M., « Using Genetic Algorithms to Learn Reactive Control Parameters for Autonomous Robotic Navigation », *Adaptive Behavior*, vol. 2, n° 3, 1994, p. pp.277-305.

[REG 02] REGGIANI M., MAZZOLI M., CASELLI S., « An Experimental Evaluation of Collision Detection Packages for Robot Motion Planning », *Conference on Intelligent Robots and Systems*, Lausanne, Switzerland, 2002.

[RUS 95] RUS D., DONALD B., JENNINGS J., « Moving furniture with teams of autonomous robots », *IROS'95*, vol. 1, 1995, p. 235-242.

[SIM 01] SIMONIN O., « Le modèle satisfaction-altruisme : coopération et résolution de conflits entre agents situés réactifs, application à la robotique », PhD thesis, Univesité Montpellier II, décembre 2001.

[STE 90] STEELS L., « Cooperation Between Distributed Agents Through Self-Organisation », DEMAZEAU Y., MÜLLER J.-P., Eds., *Decentralized A.I. : Proc. of the First European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Cambridge, England*, p. 175-196, North-Holland, Amsterdam, 1990.

[WAN 91] WANG P., « Navigation Strategies for Multiple Autonomous Mobile Robots Moving in Formation », *Journal of Robotics Systems*, vol. 8, n° 2, 1991, p. pp.177-195.

[YAM 01] YAMAGUCHI H., ARAI T., BENI G., « A distributed control scheme for multiple robotic vehicles to make group formations », *Robotics and Autonomous Systems*, vol. 36, 2001, p. pp.125-147.