Ancrage d'un lexique partagé entre robots autonomes dans un environnement non-contraint

Matthieu Nottale

UNIVERSITE PIERRE & MARIE CURIE
LA SCIENCE A PARIS

Institut Formation Doctorale

THESE DE DOCTORAT DE
L'UNIVERSITE PIERRE ET MARIE CURIE

Spécialité

Informatique, télécommunication et électronique

Présentée par

M. Matthieu Nottale

Pour obtenir le grade de

DOCTEUR de l'UNIVERSITÉ PIERRE ET MARIE CURIE

Ancrage d'un lexique partagé entre robots autonomes dans un environnement non-contraint.

soutenue le 19 Juin 2008

devant le jury composé de :

Pr. Luc Steels	(président)
Pr. Jean-Gabriel Ganascia	(directeur de thèse)
Dr. Jean-Christophe Baillie	(examinateur et encadrant)
Pr. Olivier Sigaud	(examinateur)
Dr. Pierre-Yves Oudeyer	(examinateur)
Pr. Philippe Gaussier	(rapporteur)
Pr. Peter Ford Dominey	(rapporteur)

ENSTA ParisTech

Résumé

Les Talking Heads sont une expérience de robotique développementale menée par Luc Steels et Frédéric Kaplan en 1995. Dans cette expérience, des agents apprennent un lexique de mots associés à des propriétés visuelles d'objets extraits de leurs perceptions. Cet apprentissage est réalisé par le biais de jeux de langage dans lesquels les agents s'échangent des symboles. L'interaction entre les agents est conçue de manière à ancrer ces symboles dans leur perception. Cette expérience apporte un début de réponse au problème de la sémantique des systèmes symboliques illustré par l'expérience de pensée de la chambre chinoise.

Les Talking Heads sont capables de faire converger un lexique de mots partagés au seins d'une population de plusieurs milliers d'agents de manière non supervisée, réalisant ainsi un début de solution au problème de l'émergence du langage.

Cette expérience a déjà été étendue dans plusieurs directions, par exemple en rajoutant une grammaire au lexique des agents et en étendant les perceptions des agents à des scènes montrant des objets en mouvement. Mais les restrictions que les expériences actuelles dérivant des Talking Heads posent sur l'environnement limitent certaines pistes de développement intéressantes.

Cette thèse a pour objectif de reproduire les Talking Heads en utilisant des robots autonomes et en se plaçant dans un environnement visuel non contraint: le laboratoire. L'intérêt de cette démarche est de confirmer dans un premier temps que le modèle des Talking Heads reste valide face à une perception plus complexe, pour pouvoir par la suite tester des modèles cognitifs plus complexes, donnant plus d'autonomie aux robots et qui n'auraient de sens que dans cet environnement suffisamment riche.

Nous nous intéressons dans un premier temps aux nouveaux problèmes introduits par la mobilité des robots: détection d'un autre robot, positionnement à ses côtés et pointage d'un élément de l'environnement.

Puis nous proposons un premier modèle de perception très proche des Talking Heads, utilisant des algorithmes de segmentation d'images. Ce modèle ne permet pas d'aboutir à la convergence des lexiques des agents dans ce cadre. Nous analysons les raisons de son échec, et proposons un second modèle basé sur les récentes avancées dans le domaine de la reconnaissance d'objets, utilisant des algorithmes de détection de points caractéristiques.

Ce modèle est tout d'abord testé sur une base de donnée d'images, pour vérifier sa capacité à catégoriser de manière supervisée, puis appliqué aux jeux de langage des Talking Heads. Les résultats montrent que les agents sont capables d'échanger des symboles représentant des zones de leur environnement, même si le taux de succès des jeux de langage reste encore faible.

Mots-clef: *robotique développementale, vision artificielle, ancrage, émergence du langage*

Abstract

The Talking Heads is a developmental robotics experiment conducted by Luc Steels and Frederic Kaplan in 1995, at Sony CSL lab. In this experiment, agents acquire a shared grounded lexicon of words associated with visual properties of objects in their environnement through language game interactions. The grounding of symbols in the agent's perception is ensured by the way the interactions are designed, thus providing an answer to the grounding problem formulated by Brooks and others, illustrated by the Chinese room experiment.
The Talking Heads are also capable of building a shared lexicon with a population of thousands of agents in an unsupervised way. They implement a model of how a language could emerge in a population of agents capable of communication.
This experiment has been extended in multiple areas: the lexicon has been extended with a grammar, and the static environnement has been modified to include scenes with moving objects. Yet the limited environnement is now limiting some interesting extensions of these experiments.
The aim of this thesis is to reimplement the Talking Heads using autonomous robots and using an unconstrained environment: our laboratory. This would first show that the Talking Heads models are able to scale to a more complex environment, and allow the development of future experiments with more complex cognitive models giving more autonomy to the agents and that would require this richer environnement.
We first focus on the new problems introduced by mobile robots: the detection of an other robot, the problem of positionning a robot, and the problem of pointing a region of the environnement to another robot.
We then explore a first model using image segmentation algorithms to stay very close to the original Talking Heads, and show why this model is failing to obtain lexicon convergence between the agents.
We finally introduce a second model using the state of the art in the object detection field to build object models based on the detection of recurring patterns in the environnement. We first show that this model can successfully be used for classification tasks on object databases, then apply it to the Talking Heads setup. Our results show that the agents are able to exchange symbols associated with regions of their environnement, although the overall game success rate stays low.

Keywords: symbol grounding, developmental robotics, computer vision

« The only thing I know is that I know nothing »

« There is no substitute for knowing what you are doing »

Table des matières

Chapitre 1

Introduction

Les débuts de l'Intelligence Artificielle : modéliser le mathématicien. On peut situer le début de l'intelligence artificielle (IA) au commencement de l'informatique dans les années 1950. Cet outil nouveau, capable de réaliser des traitements automatiques sur de l'information est en effet l'outil qui manquait jusqu'alors pour implémenter et tester des modèles cognitifs. Les premiers travaux tentent alors de caractériser formellement l'intelligence pour la réduire à un problème algorithmique [McCarthy and Hayes, 1969][Minsky, 1960], tout en s'interrogeant d'un point de vue plus philosophique sur le sens d'une telle démarche [Minsky, 1982]. Les premiers systèmes développés sont majoritairement symboliques, tel SOAR, toujours développé aujourd'hui [Laird and Rosenbloom, 1994] ou ACT [Anderson and Lebiere, 1998] et neuromimétiques [Rosenblatt, 1958].

Les approches symboliques ont rencontré un certain succès dans la résolution de problèmes symboliques, comme l'illustre la célèbre victoire du programme Deep Blue sur le champion du monde d'échec d'alors, Gary Kasparov, en 1997. Néanmoins ces systèmes se sont aussi heurtés rapidement à plusieurs problèmes à la fois pratiques et philosophiques.

Le premier problème porte sur la notion de sémantique et est illustré par l'expérience de pensée de la chambre chinoise [Searle, 1980] :

Un système symbolique peut donner l'impression à un observateur qu'il comprend le sens des symboles qu'il manipule. Or du point de vue de ce système seuls existent les symboles et l'ensemble des règles pour les manipuler. En conséquence, ces symboles n'ont pas de *sens* pour le système. La question de la caractérisation de ce qu'est le *sens* («the meaning of meaning») est en effet une question centrale pour l'I.A.

Le second problème plus pratique est la difficulté qu'ont rencontrés ces systèmes à s'appliquer au monde non symbolique : la difficulté à obtenir un modèle symbolique pertinent à partir de perceptions visuelles et auditives brutes, tâches que nous effectuons en permanence sans en avoir conscience, avait clairement été sous-estimée [Brooks, 1990].

L'approche non symbolique : modéliser la fourmi. Face à ces difficultés, de nombreuses architectures appelées non symboliques ou sub-symboliques ont été développées [Brooks, 1986]. Ces architectures ont en commun de ne pas essayer de construire un modèle explicite de leur environnement, suivant le principe "Le monde est son propre meilleur modèle", citation attribuée à Brooks. On pourrait placer dans cette catégorie la recherche actuelle en apprentissage par renforcement, ainsi que l'approche "animat", qui propose des modèles inspirés des systèmes biologiques capables de reproduire les comportements de certains animaux (rat, fourmi)[Doncieux and Meyer, 2003].

Cette voie de recherche est toujours en développement aujourd'hui, mais aucun modèle non symbolique n'arrive encore à expliquer ou reproduire des comportements au delà d'un niveau de complexité basique.

L'approche développementale et l'ancrage des symboles : modéliser le bébé. L'ancrage de la symbolique est une voie de recherche prometteuse qui tente de résoudre les problèmes de l'I.A. exposés ci-dessus [Harnad, 1990] : cette approche consiste à associer certains symboles à des éléments de la perception du système afin de les *ancrer* dans la réalité. La branche de recherche appelée robotique développementale explore des modèles basés sur ce principe, en s'inspirant des résultats de la psychologie développementale, branche de la psychologie qui étudie les mécanismes du développement du savoir chez les enfants à partir de la naissance.

Cette voie de recherche prometteuse n'est néanmoins pas sans écueils : [Taddeo and Floridi, 2005] font une critique pertinente de nombreux systèmes tentant de résoudre le problème de l'ancrage des symboles, en leur reprochant ce que les auteurs appellent "semantic commitment" (engagement sémantique) : le chercheur qui développe un système attribue implicitement une valeur sémantique à ces divers composantes et interprète ses résultats à partir de cette conception. Ainsi l'intelligence que semble exhiber un tel système n'est que dans l'oeil de celui qui l'observe.

Les travaux présentés ici se placent dans cette voie de recherche, de même

que les *Talking Heads* [Steels, 1998] dont ils dérivent et auquels le chapitre suivant est consacré.

Chapitre 2

Les Talking Heads

Les *Talking Heads* sont une série d'expériences démarrées par Luc Steels et Frédéric Kaplan en 1995, s'inscrivant dans la voie de recherche développementale brièvement décrite au chapitre précédent. Elles ont pour but d'étudier comment une population d'agents peut partager et faire évoluer un lexique de mots associés à des propriétés des objets de leur environnement. Les travaux que nous allons présenter dans cet ouvrage s'inspirant largement des *Talking Heads*, ce chapitre leur est entièrement consacré.

2.1 Présentation générale

L'expérience des *Talking Heads* propose un modèle tentant de répondre aux problématiques suivantes :

- Comment un agent peut-il classifier ses perceptions de son environnement de manière autonome et adaptative ?
- Comment un langage peut-il émerger dans une population d'agents ? Et plus spécifiquement, comment un agent peut-il enseigner à un autre agent une association entre un mot et une classe de perceptions (signifiant), en ne pouvant communiquer avec lui qu'à travers leur environnement ?

Le principal problème que pose la communication entre agents au moyen d'un langage est illustré fig. 2.1, appelée le carré sémiotique (*semiotic square*) : considérons la situation dans laquelle un agent souhaite désigner un objet à un autre agent au moyen d'un mot, qui est la situation utilisée dans le modèle des *Talking Heads*. Tout d'abord, cet agent n'a accès qu'à sa perception de

l'objet, qui n'est qu'une facette de celui-ci et est différente de la perception qu'a le second agent de ce même objet. Ensuite l'agent ne peut communiquer sa perception brute. Il doit passer par une phase de conceptualisation, au cours de laquelle cette perception est abstraite. Ces concepts, ou classes de perceptions, sont des éléments discrets auxquels l'agent peut attacher des mots. Ainsi, pour désigner un objet, l'agent prononce un mot associé à une abstraction qui correspond à sa perception de l'objet désigné.

Le second agent qui entend ce mot doit effectuer le cheminement inverse. Il cherche une de ses classes de perceptions associée au mot entendu, puis une de ses perceptions correspondant à cette classe, ce qui lui donne l'objet auquel se référait le premier agent.

La communication entre les agents, et donc l'apprentissage de cette communication, passe ainsi par de nombreuses étapes intermédiaires, irréductibles et propres à chaque agent.

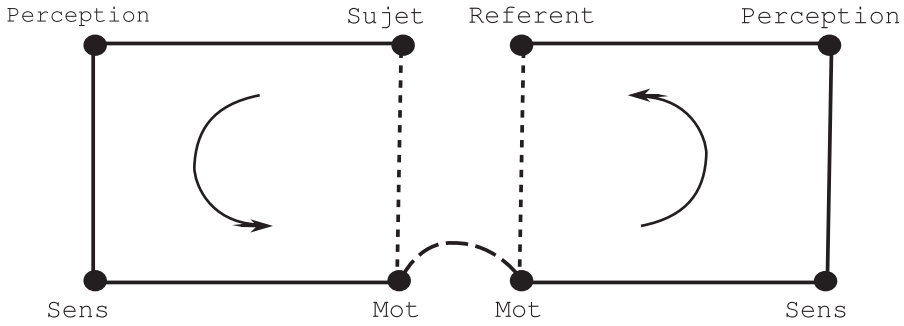


FIG. 2.1 – Carré sémiotique.

Les *Talking Heads* présentent à la fois un modèle plausible expliquant l'émergence du langage dans les civilisations humaines et un système opérationnel permettant à des agents robotiques d'apprendre un vocabulaire à l'aide d'interactions avec un humain ou un autre agent, en s'appuyant sur deux grands principes pour gérer la complexité de la tâche :

– Le *sélectionisme* du système de classification : les structures de données de classification sont en permanence modifiées par un double mécanisme de croissance et d'élagage probabiliste : les classes les plus souvent rencontrées sont étendues, pour donner de nouvelles classes plus spécialisées, et les classes les moins utiles pour l'activité de l'agent sont élaguées

afin de limiter la taille du sytème de classification et donc la puissance de calcul nécessaire pour le faire fonctionner. La valeur d'utilité d'une classe est définie comme le taux de succès des interactions la mettant en jeu. Ce double mécanisme permet de maintenir en permanence un système de classification adapté à l'environnement et à la communication.

– La *salience* des perceptions : à chaque perception est associée une valeur de salience, calculée en fonction du pouvoir de discrimination qu'elle apporte. Cette salience est utilisée par les agents au moment de l'apprentissage comme un biais permettant de réduire les ambiguïtés.

2.2 Les agents et leur environnement

Les agents des *Talking Heads* sont de simples caméras montées sur trépieds. En face d'elles est placé un tableau blanc sur lequel sont accrochées des formes géométriques colorées(cf fig. 2.2) qui constituent leur environnement.

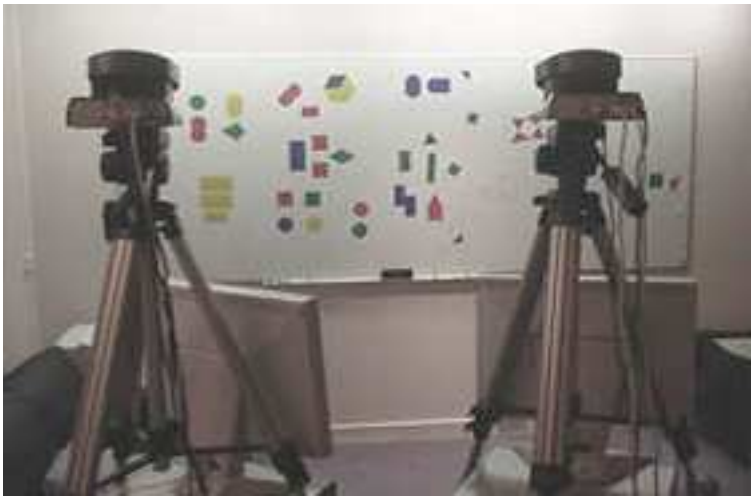


FIG. 2.2 – Les *Talking Heads* et leur environnement.

Cet environnement est à la fois suffisamment simple pour être aisément perceptible par un système de vision artificiel, dont la conception n'est pas le cœur du sujet des *Talking Heads* et suffisamment complexe et varié pour montrer l'efficacité du modèle.

Les caméras sont placées à une distance telle qu'entre 3 et 5 objets environ soient présents dans le champ à chaque itération. Ces objets sont appelés le *contexte* de l'itération. Ce dispositif permet d'obtenir un nouveau contexte simplement en réorientant aléatoirement la caméra.

2.3 Perception

Chaque agent traite une image du contexte acquise par la caméra pour obtenir un premier niveau d'abstraction de ses perceptions : tout d'abord l'image est segmentée, en fusionnant les résultats de deux algorithmes de segmentation simples, l'un détectant les contours et l'autre les régions de couleur homogène. Étant donné la simplicité de la scène et la présence d'un fond uni, cet algorithme arrive presque systématiquement à extraire tous les objets du contexte.

Ensuite, à chaque objet est associé une série de valeurs correspondant à certaines de ses propriétés, calculées par différents *canaux perceptifs* : largeur, hauteur, aire, position moyenne horizontale et verticale, luminosité moyenne, ainsi qu'un descripteur de forme. Les valeurs de chaque canal se situent toujours entre 0 et 1.

Ces valeurs sont ensuite normalisées linéairement en fonction du contexte, de sorte que, pour chaque canal, la valeur la plus basse devienne 0 et la plus haute 1. Cette étape permet d'utiliser au mieux l'espace des valeurs possibles. La perception de chaque objet est ainsi relative au contexte de cette perception et non absolue. Mais ce n'est pas gênant car, comme nous le montrerons par la suite, les communications entre *Talking Heads* se basent sur la capacité à discriminer un objet de son contexte et non sur l'identification d'un objet entre les différentes itérations.

A chacune de ces valeurs est associée une valeur de *salience*, égale à la distance à la valeur la plus proche sur le même canal perceptif, en considérant tous les objets du contexte. Plus la salience d'une valeur est importante, plus cette valeur est discriminante. Toutes les valeurs dont la salience est inférieure à un seuil fixé sont ignorées par les agents. Si un objet n'a aucune propriété saliente, c'est-à-dire dont la valeur associée est supérieure au seuil de salience, il est supprimé du contexte et ignoré par l'agent.

2.4 Signifiant, arbres de discrimination

L'objectif d'un agent des *Talking Heads* est d'acquérir un vocabulaire de mots lui permettant de désigner un objet à un autre agent, en le discriminant de son contexte. Chaque mot est pour ce faire associé à une classe délimitant un intervalle de l'espace des perceptions. Trouver un mot désignant un objet dans un contexte donné revient à trouver un mot associé à une classe correspondant à la perception de cet objet et à aucun autre objet du contexte. Les agents des *Talking Heads* doivent donc dans un premier temps discrétiser leurs perceptions, pour former des classes auxquelles pourront s'associer les mots.

Cette classification est réalisée au moyen d'arbres de discrimination (*discrimination trees*) : A chaque canal perceptif est associé un arbre binaire. Chaque nœud de l'arbre est associé à un intervalle de [0,1]. Le nœud racine est associé à [0,1] et chaque paire de nœuds fils partitionne en deux l'intervalle du nœud parent.

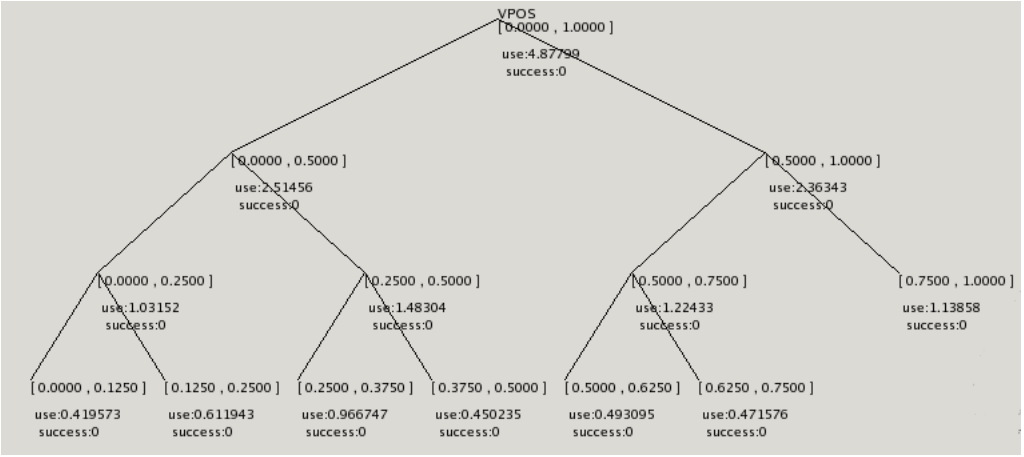


FIG. 2.3 – Exemple d'arbre de discrimination

Il est nécessaire que chaque arbre soit suffisamment développé pour que des nœuds suffisamment discriminant soient présents, mais si l'arbre est trop développé, il contient des nœuds trop spécifiques qui nuisent à l'apprentissage. Les *Talking Heads* utilisent un double mécanisme de croissance et d'élagage qui permet de maintenir les arbres de discrimination à une taille adaptée. Ce mécanisme est guidé par le taux de succès de communication entre agents

(défini dans les sections suivantes) : si celui-ci est faible, la croissance est privilégiée, considérant que les échecs sont causés par l'incapacité des agents à discriminer correctement. Si celui-ci est élevé, l'élagage est privilégié.

Le choix des nœuds à développer est fait en considérant le taux d'utilisation de chaque nœud : plus une classe est observée, plus il est intéressant de la spécialiser. Le choix des nœuds à élaguer est fait en considérant pour chaque nœud le taux de succès des interactions dans lesquelles il est utilisé : un nœud associé à un taux de succès faible a plus de chance de correspondre à une perception qu'il est difficile de partager, et donc inutile dans le contexte des *Talking Heads*.

Ce mécanisme permet aux *Talking Heads* de s'adapter aux changements dans leur environnement en adaptant leur capacité de discrimination en permanence.

2.5 Lexique

Les mots utilisés par les *Talking Heads* sont de simples combinaisons aléatoires de syllabes choisies parmi une liste fixée à l'avance. Chaque agent dispose d'un lexique, sous forme d'une liste de triplets (mot, signifiant, poids). Les signifiants sont les nœuds des arbres de discrimination. Le lexique peut bien sûr présenter des synonymes (plusieurs mots pour le même signifiant) et des homonymes (le même mot avec plusieurs signifiants différents). Lorsqu'un mot doit être choisi pour un signifiant donné, celui dont l'association avec ce mot a le poids le plus élevé dans le lexique est choisi.

2.6 Initialisation : le jeu de discrimination, le jeu de nommage

Chaque agent devant participer à l'expérience passe par une phase d'initialisation de ses arbres de discrimination, afin de ne pas débuter les interactions avec des arbres vides. Cette phase n'est pas strictement nécessaire, mais permet d'accélérer le déroulement de la phase d'interaction. Cette phase est appelée jeu de discrimination (*discrimination game*).

Chaque itération du jeu de discrimination se déroule comme suit : L'agent choisit aléatoirement un contexte, composé de 3 à 5 objets comme exposé plus haut, et extrait pour chaque objet ses valeurs sur chaque canal perceptif et les

nœuds des arbres de discrimination correspondants. Ces valeurs sont ensuite filtrées par salience et normalisées. L'agent choisit ensuite aléatoirement un objet parmi les objets du contexte, appelé le *sujet* de l'itération et tente de le *discriminer* : il cherche un nœud d'un de ses arbres de discrimination s'appliquant au sujet, mais à aucun autre objet du contexte. L'itération est un succès si un tel nœud existe, un échec sinon.

En itérant le jeu de discrimination et en utilisant le mécanisme de croissance/élagage décrit section 2.4, les arbres de discrimination de l'agent croissent puis se stabilisent une fois un certain taux de succès atteint, dont la valeur est fixée par les paramètres du système croissance/élagage.

Optionnellement, chaque agent peut aussi initialiser son lexique seul, en complétant le jeu de discrimination par une phase de nommage dans laquelle l'agent crée une entrée du lexique associant un nouveau mot choisi aléatoirement au meilleur nœud (celui associé au canal le plus salient) discriminant le sujet.

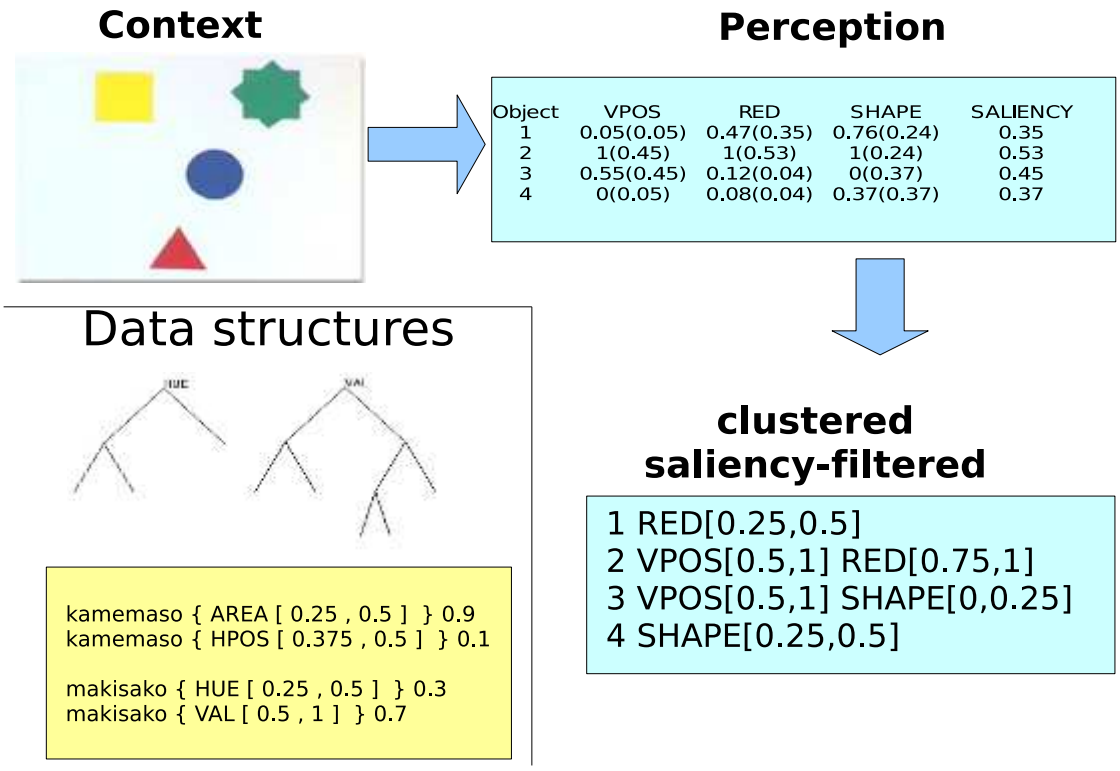


FIG. 2.4 – Déroulement de la perception des agents et structures de données

2.7 Le Guessing Game

L'échange de mots entre agents se fait par l'intermédiaire d'une interaction appelée jeu de devinette (*guessing game*). Ce jeu se déroule entre deux agents, incarnés dans deux caméras placées côtes à côte, partageant le même contexte choisi aléatoirement. Les deux agents jouant des rôles différents dans cette interaction, l'un est choisi aléatoirement pour être le leader, appelé *speaker* ; le second agent est appelé *hearer*.

Le *guessing game* débute comme le jeu de discrimination : le *speaker* choisit un des objets comme sujet de l'itération et cherche un nœud de ses arbres de discrimination le discriminant du reste du contexte. Au cas où il en existe plusieurs, le *speaker* choisit le plus générique associé au canal avec la plus haute salience. Puis il cherche le mot de son lexique avec la plus forte

association avec ce nœud, le créant si nécessaire. Le *speaker* prononce alors ce mot (en réalité dans les *Talking Heads*, les mots sont échangés sous forme de chaînes de caractères par les agents).

Le *hearer* cherche à "deviner" le sujet auquel se *réfère* le mot qu'il entend : il cherche dans son lexique le nœud ayant la plus forte association avec ce mot. Il cherche ensuite la liste des objets du contexte auxquels ce nœud s'applique. Si cette liste contient un seul élément, le *hearer* désigne cet objet au *speaker* (en lui communiquant les coordonnées de son centre dans le référentiel de l'image). Sinon, le *hearer* signale le *speaker* qu'il n'a pas trouvé de référent.

Si le *hearer* a désigné un objet, le *speaker* vérifie qu'il s'agit bien du sujet. Si c'est le cas, il signale l'information au *hearer* et cette itération du *guessing game* est un succès. Dans les autres cas (si le *hearer* n'a rien pointé, ou a pointé un autre objet), le *speaker* pointe le sujet au *hearer* et l'itération est un échec.

En cas d'échec, le *hearer* tente d'apprendre une nouvelle association mot-nœud à partir de l'information communiquée par le *speaker* : un objet et un mot. Pour ce faire, il doit deviner le signifiant choisi par le *speaker* pour désigner ce mot parmi tous les nœuds possibles, c'est-à-dire ceux qui discriminent cet objet des autres. Le *hearer* choisit d'associer le signifiant le plus générique sur le canal le plus salient discriminant le sujet au nouveau mot. Ce choix permet d'obtenir une association "correcte" si l'environnement est perçu de manière suffisamment similaire par les deux agents.

Après chaque itération du jeu, les deux agents mettent à jour leurs structures de données : les arbres de discrimination sont élagués et augmentés comme décrit section 2.4 et les poids des mots des lexiques sont mis à jour comme suit : en cas de succès, le poids de l'association utilisée est augmenté d'une constante par les deux agents. Le *speaker* réduit aussi d'autant le poids de tous les synonymes et le *hearer* le poids de tous les homonymes. En cas d'échec, le poids de l'association utilisée est diminué de cette même constante par les deux agents.

```
Game 101
a4 is the speaker, a1 is the hearer
a4 segments the context into 2 objects : o1 and o2
a4 choses o1 as the topic
a4 categorises the topic as [GRAY 0.5 :1]
a4 has two words for [GRAY 0.5 :1] :
- faluleru (0.2)
- notabefe(0.0)
a4 says faluleru
a1 does not know faluleru
a1 says faluleru ?
a4 points o1
a1 categorises the topic as [GRAY 0.5 :1]
a1 stores faluleru as [GRAY 0.5 :1]
```

FIG. 2.5 – Exemple de déroulement du *guessing game*.

2.8 Résultats, extensions et travaux connexes.

Pour des populations d'une dizaine d'agents, la convergence des lexiques est atteinte en quelques milliers d'itérations [Steels, 1998]. Le taux de succès du *guessing game* se maintient ensuite indéfiniment entre 0.9 et 1, les arbres de discrimination restant à une taille constante.

[Steels, 1999] étudie plus en détail l'évolution des synonymes dans le lexique et montre que, lorsque plusieurs mots sont en compétition pour un même sens dans une population, l'un d'eux finit toujours par l'emporter en effaçant complètement les autres.

[Vogt and Coumans, 2003] et [Vogt and Divina, 2005] étudient les problèmes posés par des grandes populations de plusieurs milliers d'agents et montrent que le lexique peut rester stable même dans un modèle dans lequel la population d'agents se renouvelle.

P. Vogt étudie dans [Vogt, 2000] un modèle d'apprentissage un peu différent, appelé *observational game*. Dans ce modèle le *speaker* pointe systématiquement le sujet en produisant un mot, et le *hearer* apprend l'association entre le mot et le signifiant le plus salient discriminant le sujet, sans aucun retour du *speaker*. L'évolution des lexiques se fait donc sans aucune information sur l'efficacité du jeu de langage. P. Vogt montre que ce modèle obtient des

résultats similaires au *guessing game*. Ses résultats montrent aussi l'importance de l'inhibition latérale lors de la mise à jour du poid des entrées du lexique pour réduire le taux de synonymie et d'homonymie des lexiques.

Dans [Vogt, 2004], l'auteur montre que les agents d'une expérience similaire aux *Talking Heads* respectent la loi de Zipf-Mandelbrot, observée en linguistique et portant sur la courbe de fréquence d'utilisation des mots du lexique.

Dans ses travaux, P. Vogt utilise un modèle de discrimination des perceptions généralisant celui des *Talking Heads* : les arbres de discrimination sont remplacés par une segmentation de Voronoï hiérarchique autour d'échantillons de la perception. Ce modèle permet d'utiliser des canaux perceptifs multidimensionnels.

[Vogt, 2005] ajoute aux agents la capacité de générer et d'apprendre une grammaire, permettant de générer de nouveaux mots composés à partir des mots élémentaires des *Talking Heads*.

[Steels and Baillie, 2003] étendent les *Talking Heads* à la description d'actions observées par les agents dans des séquences vidéos.

[de Jong E., 1998] développe un système dans lequel les agents utilisent le langage qu'ils développent pour se signaler la présence de prédateurs.

[Michael, 2002] utilise des jeux de langages similaires à ceux des *Talking Heads*, dans un monde virtuel dans lequel les perceptions des agents sont tirées d'une distribution de type somme de gaussiennes. Chaque agent classifie son environnement indépendamment des jeux de langages. Ce modèle lui permet de contrôler les "concepts" présents dans les perceptions des agents et d'étudier l'impact du nombre de concepts et du taux de recouvrement sur les jeux de langage.

[Ikegami and Iizuka, 2007] illustre l'importance de la prise de rôle différents (enseignant-élève) en alternance pour l'apprentissage, sur un problème d'apprentissage de mouvements.

Chapitre 3

Les Talking Robots

3.1 Cadre

Les modèles introduits par les *Talking Heads* ont été largement utilisés, testés et étendus [Steels, 2007]. Toutes ces expériences se limitent soit à des mondes simulés, soit à des environnements très simples, les problèmes de visions associés à un environnement plus complexe n'étant pas au centre de leurs objectifs.

Mais cette simplicité de l'environnement limite aussi la complexité de ce que les agents peuvent apprendre : dans un monde se réduisant à des formes géométriques, l'agent ne peut faire plus qu'apprendre à les classifier. De plus, ces expériences ne donnent pas la possibilité aux agents d'agir sur leur monde, où, lorsque ils agissent, ils ne font qu'exploiter ce qu'ils ont appris : l'action ne sert pas à l'apprentissage.

Des progrès récents dans deux grands domaines permettent d'envisager de pousser les *Talking Heads* plus loin, vers l'apprentissage de concepts "utiles" pour percevoir et agir dans le monde des Humains [Kaplan, 2000] :

– La robotique : depuis quelques années des robots mobiles autonomes et programmables disposant de caméra sont disponibles à un prix raisonnable dans le commerce. Citons par exemple les aibo de Sony, introduits en 1999. Utiliser ces plateformes facilement accessibles et fiables évite aux laboratoires qui le souhaitent de consacrer des années au développement d'une solution équivalente. L'arrivée du langage de programmation et *middleware* Urbi [Baillie, 2005], spécialement conçu pour la robotique, permet de plus un développement rapide sur ces plateformes, rendant

l'expérimentation plus aisée.
– La vision artificielle : Les algorithmes de cartographie comme le SLAM [Davison, 2003] peuvent permettre de faire se déplacer intelligemment un robot mobile dans son environnement. Les algorithmes de détection de points caractéristiques robustes et tournant en temps réel comme SIFT [Lowe, 2004] constituent une nouvelle base solide pour effectuer de l'apprentissage et de la reconnaissance d'objets, même de manière non supervisée.

3.2 Objectifs

Partant de ce constat, l'objectif donné à cette thèse consiste à produire des expériences similaires aux *Talking Heads* dans leurs principes et dans leurs objectifs, en en changeant deux éléments importants :

3.2.1 Vers des agents mobiles, autonomes et incarnés

Tout d'abord, les agents des *Talking Heads* (des caméras sur un trépied) seront remplacés par des robots mobiles autonomes. La mobilité des agents leur permettra d'accéder à un environnement plus varié. Rappelons que c'est la richesse de l'environnement, à savoir le fait qu'un même objet puisse être vu dans des contextes différents qui permet aux agents des *Talking Heads* de résoudre les ambiguïtés. Cette mobilité pourra aussi permettre à terme d'intégrer un modèle de curiosité à l'apprentissage, dans lequel l'agent sélectionne l'environnement dans lequel il apprend en fonction de ses capacités [Oudeyer et al., 2007]. Cette mobilité ouvre aussi la possibilité à l'agent de manipuler son environnement dans de futures expériences. Nous essayerons de donner à ces agents la plus large autonomie possible et de limiter le nombre d'interventions humaines nécessaires au déroulement des jeux de langage.

Nous avons choisi d'utiliser des Aibo ERS7 de Sony. Il s'agit de robots quadrupèdes disposant de capteurs de distance, d'une caméra de résolution 208x160, d'un processeur MIPS cadencé a 400MHz, d'une interface WiFi et de 64Mb de RAM. Leur autonomie énergétique est d'environ 1h.

3.2.2 Vers un environnement peu restreint

Le point central des travaux présentés ici consiste à remplacer l'environnement contraint des *Talking Heads* par l'environnement "naturel" de nos robots : le laboratoire. L'objectif de nos robots est similaire à celui des agents des *Talking Heads* : classifier leurs perceptions pour en extraire des "concepts" et se constituer un vocabulaire de mots associés à ces concepts, partagé avec les autres robots. La dynamique des jeux de langage des *Talking Heads* est telle que, si un lexique partagé émerge, les concepts associés à ses mots seront aussi partagés entre les robots et potentiellement utilisables pour s'échanger des informations utiles à de futures interactions.

La question principale associée à cette extension est la notion d'*objet* qui sera utilisée par les robots. A quelle structure issue de la perception les robots vont-ils associer des concepts ? Quels algorithmes utiliser pour obtenir des concepts stables et robustes, de manière non supervisée ?

3.3 Déroulement

Nos expérimentations nous ont conduit à proposer deux réponses successives à cette question.

Dans un premier temps, nous avons choisi de rester aussi proche que possible des *Talking Heads*. Un objet pour un agent des *Talking Heads* est une région de l'image perçue, donnée par un algorithme de segmentation, et un concept est un sous-ensemble des valeurs possibles (une classe) pour une des propriétés visuelles de cette région. Notre première version des *Talking Robots* procède de même, en utilisant un algorithme de segmentation d'image plus robuste pour pallier l'absence de fond homogène. Le chapitre 4 est consacré à cette première version des *Talking Robots.*

Nous n'avons pas réussi à obtenir convergence des lexiques avec ce modèle : une segmentation ad-hoc d'un environnement sans contrainte se révèle ne pas être assez robuste pour que deux robots côtes à côtes et regardant dans la même direction obtiennent des perceptions suffisamment similaire. Et sans perceptions partagées, aucune communication n'est possible.

Cet échec nous a conduit à proposer un nouveau modèle s'inspirant de la littérature en vision artificielle sur l'apprentissage et la reconnaissance non supervisée d'objets. Ce modèle se base sur des algorithmes de détection de points caractéristiques pour construire et reconnaître des modèles d'objets

dans des images. Les concepts auxquels les agents associent des mots sont les modèles d'objets que chaque agent construit indépendamment. Contrairement aux *Talking Heads*, les concepts ne sont plus partagés entre les agents, mais le principe et l'intérêt du jeu de langage reste le même : si les agents sont capables de mettre en correspondance leurs concepts d'objets au travers d'un lexique, alors ces concepts sont pertinents et utiles pour représenter leur environnement.

Le chapitre 5 est consacré à cette seconde version des *Talking Robots* : le modèle d'apprentissage est d'abord validé sur des bases de tests, à la fois de manière supervisée et non supervisée, puis utilisé dans le jeu du *guessing game*. Les résultats obtenus montrent un taux de succès significatif du jeu de langage, mais le lexique reste très bruité. Diverses pistes pour améliorer la robustesse du système sont explorées.

Chapitre 4

Talking Robots 1.0, segmentation a priori

4.1 Présentation générale

Le présent chapitre est consacré à la première version des *Talking Robots* que nous avons développée, avec pour objectif de rester aussi proche que possible des *Talking Heads*. Nous n'avons pas réussi à obtenir la convergence des lexiques, même en nous limitant à deux agents, avec cette implantation. Néanmoins ces résultats méritent d'être détaillés, car ils mettent en lumière les pré-requis nécessaires aux mécanismes d'apprentissage des *Talking Heads*.

La première partie de ce chapitre est consacrée aux problèmes nouveaux dus à l'utilisation de robots autonomes et aux solutions que nous y avons apportées : localisation des robots, synchronisation entre les agents et pointage d'objets. La seconde partie concerne les modifications des mécanismes des *Talking Heads* nécessaires pour les adapter au nouvel environnement. Le reste du chapitre expose les résultats obtenus.

4.2 Mécanismes auxiliaires

4.2.1 Synchronisation

Les interactions des *Talking Robots* présentent de nombreux points de synchronisation : par exemple, lorsqu'un robot désigne un objet, il doit prévenir l'autre robot du succès ou de l'échec de cette opération. Le deuxième robot

doit ensuite indiquer au premier s'il a réussi à trouver l'objet désigné.

Il nous faut un mécanisme simple et très robuste, utilisant les actionneurs et capteurs des aibos conformément à nos objectifs. Nous avons opté pour l'émission de sons mono-fréquence d'une durée de quelques secondes. Ceux-ci sont détectés par une simple transformée de Fourier sur une fenêtre de 20 millisecondes : le signal est reconnu si la composante principale de la transformée de Fourier est la fréquence attendue avec un rapport signal sur bruit suffisant pendant 3 fenêtres successives. Ce processus s'est révélé très robuste même en présence de bruit.

4.2.2 Synthèse et reconnaissance vocale

L'approche conforme à nos objectifs permettant aux robots de s'échanger des mots serait d'utiliser de la synthèse et de la reconnaissance vocale. Ce travail a été commencé par un stagiaire de DEA, mais n'est pas encore intégré a l'expérience des *Talking Robots*.

Pour le moment les mots sont échangés directement par le biais de messages réseaux UDP, avec optionnellement une probabilité de déformation afin de simuler des erreurs de reconnaissance.

4.2.3 Pointage

4.2.3.1 Présentation du problème

Les *Talking Robots* ont besoin d'être capables de désigner une zone de leur environnement visuel. Cette désignation doit pouvoir être interprétée par un autre agent ayant un point de vue différent sur la scène.

Les *Talking Heads* utilisaient un simple passage de coordonnées 2D, vraisemblablement corrigées pour prendre en compte l'écart de point de vue entre les deux agents. Nous ne pouvons pas utiliser cette méthode, parce que nous ne connaissons pas a priori l'écart de position entre nos deux agents, que leur environnement n'est pas plan et que nous ne disposons d'aucune information tridimensionnelle.

Un analogue d'une main qui pointe vers l'objet est difficilement réalisable, principalement parce que les aibo ne disposent pas de la stéréo-vision qui serait nécessaire pour calculer la direction pointée.

Une autre option envisageable serait de transmettre une fenêtre de l'image autour de la zone désignée et d'utiliser un algorithme de recalage pour

rechercher cette zone dans l'environnement de l'autre agent. Mais les différences entre les perceptions des deux agents dans un environnement présentant du relief rendent le recalage difficile.

L'option restante consiste à donner un moyen à l'agent de modifier sa perception (et donc celle de l'autre robot) de la zone désignée, par exemple en modifiant sa luminosité.

Nous avons pour cela placé une diode laser de 5mW sur la patte avant de chacun des aibos. Cette diode est interfacée au robot à travers une photo-résistance placée sur le dos de l'aibo, au-dessus d'une de ses LEDs, comme indiqué figure 4.1. Ce montage permet à l'aibo de commander l'allumage du laser en commandant les LEDs de son dos.

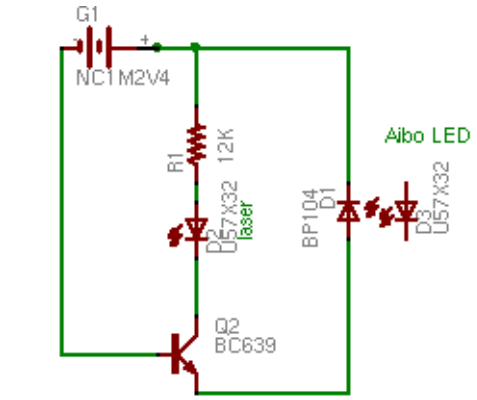


FIG. 4.1 – Schéma électrique du dispositif de pointage

Reste à résoudre le problème de la détection du spot laser. Celui-ci est très visible à l'oeil nu, même à 5 mètres et sur des surfaces sombres, mais beaucoup moins à la caméra de l'aibo, de résolution 208x160. Les sections suivantes présentent deux algorithmes de détection de spot laser et comparent leurs performances. Les deux algorithmes font clignoter le laser, le premier à une fréquence fixe connue, le deuxième le plus rapidement possible et exploitent les différences entre images successives. La difficulté réside dans le fait que la différence d'intensité lumineuse perçue par le robot entre laser allumé et laser éteint est du même ordre de grandeur que la variation d'intensité due au bruit dans les configurations les plus défavorables.

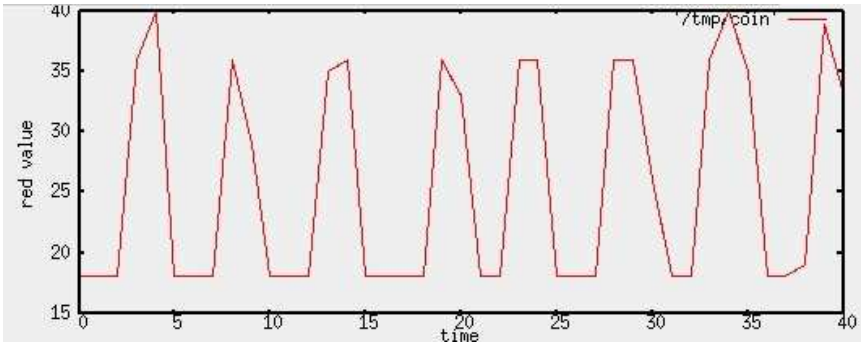


FIG. 4.2 – Variation de l'intensité du canal rouge de la caméra de l'aibo sur le pixel ou pointe le laser.

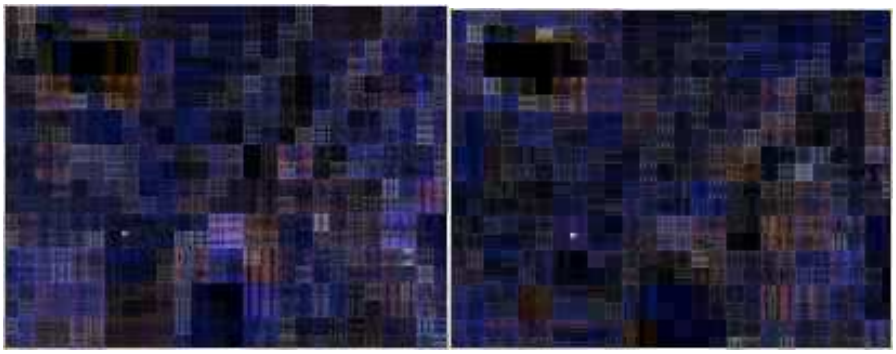


FIG. 4.3 – Agrégation normalisée de la valeur absolue de la dérivée temporelle sur une seconde, en lumière naturelle et artificielle. Le point le plus lumineux est le spot laser clignotant.

4.2.3.2 Premier algorithme : modélisation et agrégation

La variation d'intensité due au laser étant du même ordre de grandeur que la variation due au bruit si la source de lumière est naturelle, ce qui les distingue est la régularité de la variation due au laser. Notre premier algorithme essaye donc de reconnaître le signal créneau correspondant au clignotement du laser dans toutes les régions de l'image dont l'intensité varie plus qu'un seuil fixé au cours du temps.

L'algorithme suppose que le laser clignote a une fréquence précisément fixée. En pratique, nous utilisons une période de 640 millisecondes, la plus faible pouvant être maintenue de manière stable par le montage.

L'algorithme procède par réductions et agrégations successives : dans une première phase, l'algorithme utilise la dérivée temporelle des images pour lister tous les pixels présentant une variation d'intensité sur le canal rouge supérieure à un seuil, fixé expérimentalement juste au dessus du niveau de bruit moyen. La valeur du canal rouge sur chacun de ces pixels est ensuite enregistrée sur deux périodes de clignotement du laser. Le stockage se fait à taux d'échantillonage fixe : a chaque nouvelle image, associée à sa date d'enregistrement, les échantillons précédant cette date et manquant sont remplis par interpolation linéaire. Ensuite, le signal sur chaque pixel est comparé au signal créneau théorique attendu en utilisant le SPOMF [Chen et al., 1994], une technique de recalage donnant un score de corrélation et un déphasage. Un score est associé à chacun de ces pixels, proportionnel au taux de corrélation et inversement proportionnel à l'erreur de phase par rapport à la moyenne des phases obtenues jusqu'alors. Ce score est ajouté aux scores déjà obtenus aux mêmes coordonnées.

Le processus est répété jusqu'à ce qu'un des scores dépasse un seuil fixé. Les coordonnées associées à ce score sont alors considérées comme contenant un pointeur laser.

Ce processus, certes complexe, présente l'avantage d'avoir un taux extrêmement faible de faux positifs et un bon taux de détection une fois les paramètres réglés. Le temps de détection est relativement long, allant de 5 à 20 secondes selon la surface et les conditions d'éclairage.

4.2.3.3 Second algorithme : vitesse

Notre deuxième algorithme est beaucoup plus rapide, mais moins robuste, et fait l'hypothèse qu'il n'y a aucun mouvement dans la scène perçue par l'aibo.

Il ne fait pas d'hypothèse sur la fréquence de clignotement du laser, celle-ci est réglée à la vitesse maximale permise par le montage physique (de l'ordre de 3 Hz). La valeur absolue de la dérivée temporelle du canal rouge est intégrée au cours du temps. Si la valeur maximale sur l'image est supérieure à la valeur du second maximum (en masquant une région de 10 pixels de côté autour du premier maximum) multipliée par un facteur (expérimentalement déterminé à 1.2), alors l'algorithme renvoie les coordonnées du maximum. Cet algorithme est plus rapide, typiquement 1 à 2 secondes, mais plus sensible aux faux positifs.

4.2.3.4 Performances et jeu de pointage

Afin d'évaluer et de comparer objectivement les performances des deux algorithmes, nous avons conçu une interaction simple entre deux agents, sur le modèle du *guessing game*, appelée le jeu de pointage. Le premier agent pointe le laser aléatoirement dans son champ de vision. Le second cherche le spot laser, signale le premier agent lorsqu'il l'a trouvé et pointe au même endroit à son tour. Si le premier agent trouve le spot laser du second et si ce spot est à l'endroit où il avait pointé, l'itération est un succès.

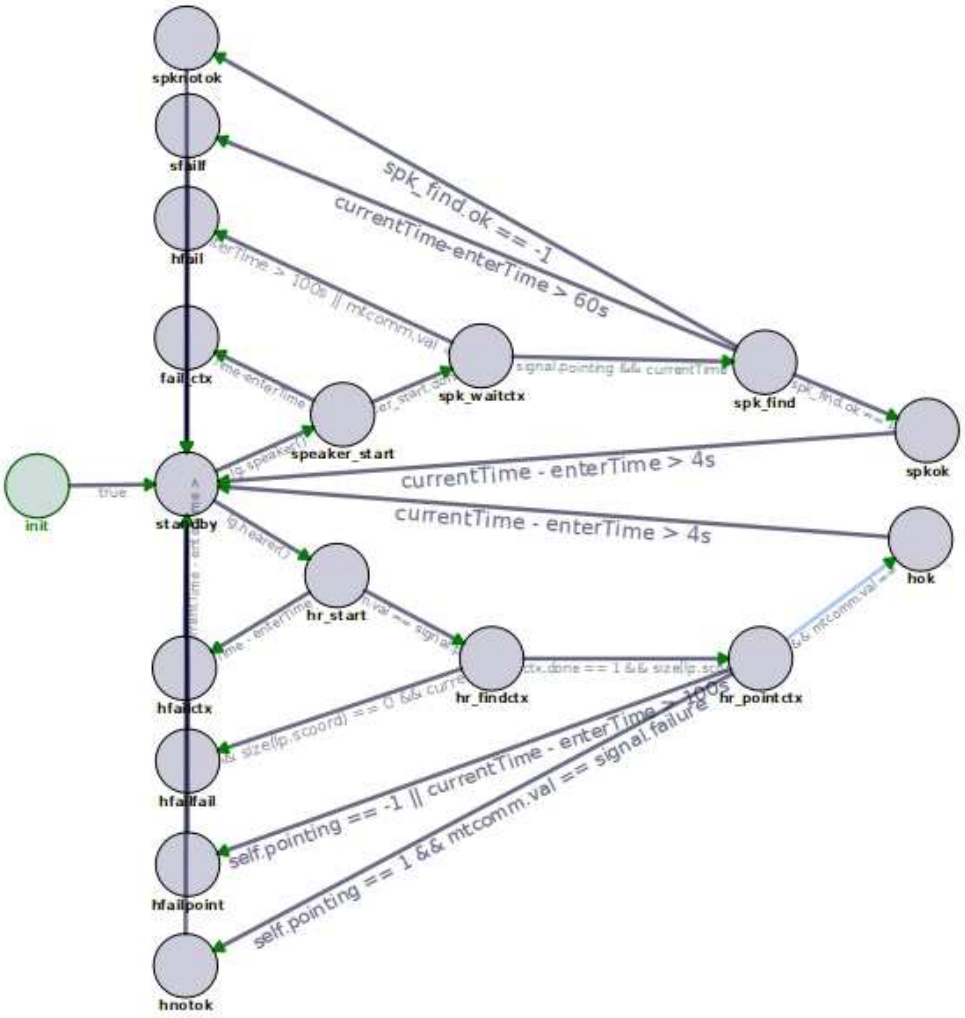


FIG. 4.4 – Graphe du comportement du jeu de pointage (application URBIDev)

En éclairage artificiel, les deux algorithmes ont un taux de succès de l'ordre de 0.8. Les échecs surviennent principalement quand le *speaker* pointe un endroit invisible pour le *hearer*, car un autre objet lui bloque la vue et quand le spot laser tombe sur une surface réfléchissante, ou très absorbante.

En éclairage naturel, c'est-à-dire quand le bruit est plus important, les performances du premier algorithme restent les mêmes et celles du second s'écroulent. Le premier exploite en effet plus d'information sur le signal du

spot laser, qui lui permettent de le distinguer du bruit, au prix d'un temps de détection plus long.

4.2.4 Localisation des aibos

Comme nous l'avons dit plus haut, les *Talking Robots* doivent être capables de se déplacer dans leur environnement, de détecter un autre robot et de se positionner à ses cotés afin de partager le même champ de vision. La difficulté de cette tâche peut se réduire à détecter la position d'un autre robot et son orientation. C'est un cas particulier du problème de reconnaissance des formes, assez proche du problème de reconnaissance de visages, qui a été très largement étudié [Hjelmas and Low, 2001]. Dans cette partie, nous exposons deux méthodes que nous avons développées, appliquées aux robots aibo. La première consiste à utiliser des méthodes de recalage et de reconnaissance d'images en utilisant la transformée de Fourier, appliquées à la tête de l'aibo. La seconde plus simple, mais plus contraignante, utilise des marqueurs bicolores placés sur l'aibo.

Le lecteur pressé pourra sauter cette section assez technique et éloignée du sujet principal sans nuire à la compréhension générale.

4.2.4.1 SPOMF et FMI-SPOMF

L'aibo est une cible particulièrement difficile à détecter : sa surface unie blanche et noire, très réfléchissante rend inadaptées les méthodes par histogramme de couleur ou d'intensité, les méthodes à base de texture, ainsi que les méthodes à base de points caractéristiques. Nous nous sommes donc orientés sur les méthodes globales.

Dans notre configuration, le deuxième robot peut aider le premier à le trouver, par exemple en hochant la tête. En supposant qu'il s'agit du seul objet mobile de la scène, le *speaker* a déjà une idée approximative de la position du *hearer*. De plus, la dérivée temporelle du flux vidéo montre le mouvement de la tête sur un fond noir et peut être utilisée pour effectuer un recalage en translation et en échelle avant d'appliquer l'algorithme de reconnaissance, voire même être utilisée directement comme motif par ces algorithmes.

Dans la partie suivante, nous détaillons le processus d'acquisition et d'apprentissage du modèle, les trois algorithmes de reconnaissance des formes que nous avons testés : corrélation croisée, SPOMF (Simple Phase-Only Matched

Filtering) et FMI-SPOMF (SPOMF of the Fourier-Mellin invariant), avant d'exposer les résultats obtenus.

Apprentissage. Le but de la phase d'apprentissage est d'acquérir une image et une image dérivée représentatives d'une tête d'aibo oscillante vue sous différentes orientations.

Les deux robots sont placés l'un en face de l'autre à une distance connue (40 centimètres), dans une position fixée et connue. Pour permettre un apprentissage automatisé, le *speaker* tourne sa tête pour que le *hearer* obtienne des images de la tête sous différents angles. Le *speaker* tourne sa tête du premier angle pour lequel on souhaite obtenir un motif et oscille autour de cette position à une fréquence de 2Hz et une amplitude de 5 degrés. L'étape suivante consiste à trouver la région de l'image contenant la tête et est décrite figure 4.5. A chaque image reçue, les N dernières images dérivées sont moyennées. Le résultat est seuillé et passé par une fermeture morphologique. On isole ensuite les régions connexes, puis on calcule les coordonnées de leurs boîtes englobantes, qu'on agrège avec les boîtes englobantes obtenues aux itérations précédentes : quand deux boîtes sont proches, c'est-à-dire quand leurs centres et leurs tailles sont à une distance relative inférieure à un seuil donné, la plus grande est conservée et son score est incrémenté. Dès qu'une boîte dépasse un score fixé, elle est conservée pour la suite comme la région contenant la tête du robot. Cette phase complexe permet de repérer correctement la région de la tête même en présence d'autres objets se déplaçant sporadiquement dans le flux d'images.

Ensuite, un ensemble d'images et d'images dérivées dans cette région est enregistré : à intervalles aléatoires, les N dernières images et images dérivées sont moyennées et la partie correspondant à la région de la tête est conservée, jusqu'à ce que suffisamment d'images aient été acquises (typiquement une vingtaine). Pour extraire une image représentative de chaque ensemble, on construit la matrice des distances entre toutes les paires d'images, en utilisant le rapport signal pic sur bruit (PSNR) du SPOMF comme fonction de distance (cette mesure est détaillée plus bas). L'image dont la somme des distances à toutes les autres est la plus basse est conservée.

Ce processus est répété pour différentes valeurs d'orientation de la tête. On aboutit à un ensemble de paires d'images et d'images dérivées, associées à une valeur d'angle jouant le rôle de descripteur de classe. Le but du filtre moyen temporel est de diminuer la variabilité des images provoquée par les

oscillations de la tête. Ce filtre devra être appliqué de manière identique dans le processus de reconnaissance.

Il convient de noter que lorsque l'algorithme sera utilisé, le *speaker* aura toujours sa tête orientée droite, mais sera positionné de manière quelconque par rapport au *hearer* et donc vu sous une orientation quelconque : les images acquises pendant l'apprentissage ne correspondent pas exactement à la situation d'utilisation, mais cela n'a pas d'influence car l'axe de rotation de la tête est perpendiculaire au sol.

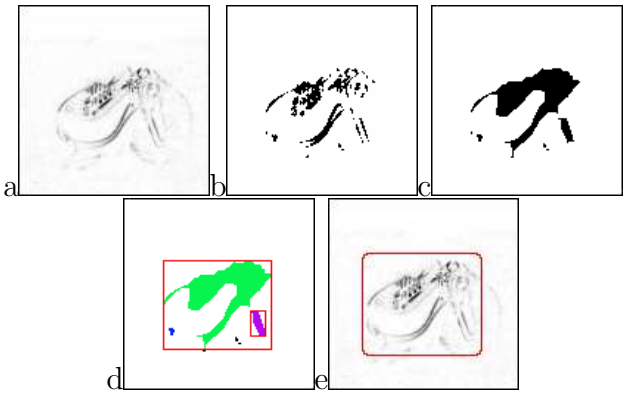


FIG. 4.5 – Recherche de la région contenant la tête : a : l'image dérivée. b : seuillage. c : fermeture morphologique d : boîtes englobantes. e : fusion des boîtes englobantes

Reconnaissance. Le modèle de reconnaissance a pour but de fournir à partir d'un flux d'images la distance, la position et l'angle d'orientation de toutes les têtes de robots trouvées, potentiellement associées avec un facteur de croyance. Les deux robots sont immobiles pendant cette phase, mis à part le mouvement d'oscillation de la tête du *speaker*.

Dans un premier temps, le modèle cherche à trouver les régions de l'image pouvant contenir une tête de robot, en utilisant un processus similaire à la phase d'apprentissage. Les N dernières images acquises sont moyennées, le résultat est seuillé et fermé morphologiquement, les régions connexes sont isolées. Chaque boîte englobante d'une région connexe est ensuite testée : la portion correspondante de l'image ou de l'image dérivée est d'abord recalée en translation et en échelle en utilisant l'algorithme détaillé dans la section suivante, puis le résultat est appliqué à l'algorithme de reconnaissance (SPOMF, FMI-SPOMF ou cross-corrélation), qui donne un vecteur de score (un score

par motif, soit par angle). Ce vecteur de score est agrégé avec les vecteurs obtenues aux images précédentes pour les régions qui concordent. Quand une région a été considérée un nombre suffisant de fois, elle est marquée comme potentiellement contenant une tête d'aibo, dont l'orientation est l'angle dont le score est le plus élevé dans le vecteur de scores.

Nous avons testé trois algorithmes de reconnaissance : corrélation croisée, qui nécessite un recalage en échelle et en translation avant d'être appliqué, SPOMF, qui nécessite un recalage en échelle et FMI-SPOMF, qui ne nécessite aucun recalage. Nous avons testé ces algorithmes sur les images d'origine et les images dérivées. Dans les parties suivantes, nous détaillons l'algorithme de recalage, puis les algorithmes de reconnaissance et enfin quelques filtres de pré-traitement que nous avons tenté d'appliquer pour améliorer les résultats.

Recalage. L'algorithme de corrélation croisée nécessite un recalage en translation et en échelle, et le SPOMF nécessite un recalage en translation. De nombreux algorithmes ont été proposés pour résoudre ce problème [Zitova and Flusser, 2003]. Nous avons opté pour un algorithme simple basé sur un calcul de moments [Werman and Weinshall, 1995] et utilisant les images dérivées, qui ont la propriété de ne pas avoir de fond.

Étant donnée une image $I(x,y)$, son centre de masse G est donné par :

$$G = \left(\frac{\sum_{x,y} I(x,y)\, x}{\sum_{x,y} I(x,y)}, \frac{\sum_{x,y} I(x,y)\, y}{\sum_{x,y} I(x,y)} \right) = (G_x, G_y)$$

et les moments $M_{0,2}$ et $M_{2,0}$par :

$$M_{2,0} = \frac{\sum_{x,y} I(x,y) \left| x - G_x \right|^2}{\sum_{x,y} I(x,y)}$$

$$M_{0,2} = \frac{\sum_{x,y} I(x,y) \left| y - G_y \right|^2}{\sum_{x,y} I(x,y)}$$

Deux images de paramètres $(G, M_{0,2}, M_{2,0})$ et $\left(G', M'_{0,2}, M'_{2,0}\right)$ peuvent être recalées en translatant la deuxième par le vecteur $\vec{G'G}$ et en multipliant l'échelle par $\left(\sqrt{\frac{M_{2,0}}{M'_{2,0}}}, \sqrt{\frac{M_{0,2}}{M'_{0,2}}} \right)$.

Dans notre cas, les motifs sont tous recalés entre eux dans une phase d'initialisation. Une image ne nécessite donc qu'un seul recalage pour être traitée, au lieu d'un recalage par motif.

Reconnaissance par corrélation croisée. Les deux formules que nous avons utilisées pour le calcul de la corrélation croisée de deux images I et I' sont celles basées sur la norme L_1 :

$$CC_{L1} = \sum_{x,y} |I(x,y) - I'(x,y)|$$

et une version simplifiée de la formule classique :

$$CC = \sum_{x,y} I(x,y)I'(x,y)$$

Cette dernière formule devrait être normalisée par le produit des variances, mais comme dans notre cas les images sont très similaires, cette étape peut être supprimée pour diminuer le temps de calcul. Comme le processus de recalage n'est pas très précis, le calcul de corrélation croisée est effectué en essayant toutes les translations possibles dans une petite fenêtre (5 par 5) et le meilleur score est conservé.

Reconnaissance par le SPOMF. L'algorithme SPOMF (*Symmetric Phase-Only Matching Filter*) utilise la différence de phase entre les transformées de Fourier de deux images pour les recaler en translation. En utilisant le rapport signal pic sur bruit (*Peak Signal to Noise Ratio*), il peut être utilisé comme algorithme de reconnaissance.

Étant donnée une image I et un motif P, on calcule leurs transformées de Fourier R et S et :

$$Q(u,v) = \frac{R(u,v)S(u,v)*}{|R(u,v)|\,|S(u,v)|}$$

où le symbole $*$ représente l'opérateur de conjugaison. Le SPOMF S est défini comme la transformée de Fourier inverse de Q. Autrement dit, l'algorithme consiste à calculer la transformée de Fourier de I et P, à calculer leur différence de phase, puis la transformée de Fourier inverse de cette différence. On peut démontrer que si I est une version translatée de P, le SPOMF est un Dirac aux coordonnées de la translation.

Pour utiliser le SPOMF comme algorithme de reconnaissance, il suffit de calculer le PSNR définit par :

$$PSNR = \frac{\left(max_{x,y}\left(S(x,y)\right) - \bar{S}\right)^2}{\frac{1}{N}\sum_{x,y}\left(S(x,y) - \bar{S}\right)^2}$$

où N est la taille du SPOMF. Dans nos expériences, un PSNR supérieur à 100 signifie que I est probablement une version translatée de P. Nous aurions pu directement utiliser la valeur du pic du SPOMF, mais les résultats sont moins marqués. La figure 4.7 montre un exemple de SPOMF sur des images dérivées d'une tête d'aibo. Le pic du SPOMF est bien visible autour de $(0, 0)$ pour le motif d'angle proche de l'image et s'amenuise au fur et à mesure que l'écart entre l'angle du motif et celui de l'image augmente.

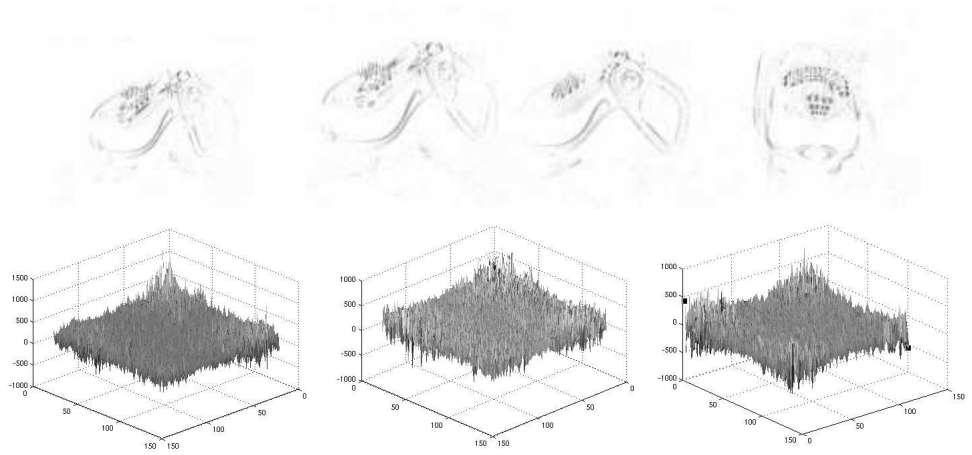


FIG. 4.7 – Image à -51°, motifs pour les angles -50, -70 et 10. SPOMF entre la premiere image et les trois motifs. Les PSNR valent respectivement 267, 94 et 33

Dans notre cas, l'objet comparé au motif par le SPOMF est souvent agrandi par l'algorithme de recalage. Il faut donc éliminer de la transformée de Fourier les composantes de haute fréquence qui n'ont pas de sens pour l'objet. Pour ce faire, la transformée de Fourier est tronquée, en utilisant la taille initiale de l'objet pour déterminer la fréquence la plus élevée qui sera conservée, avant de calculer la différence de phases et la transformée de Fourier inverse de cette dernière. On montre que si l'objet était de taille N, seule la partie centrale de taille NxN de la transformée de Fourier doit être conservée.

Reconnaissance par le FMI-SPOMF. L'algorithme FMI-SPOMF consiste à appliquer le SPOMF sur l'invariant de Fourier-Melin (FMI) et peut être utilisé pour recaler deux images en rotation, translation et échelle [Chen

et al., 1994].

Le passage en coordonnées log-polaires a la propriété de transformer les rotations et les changements d'échelle en translations [Wolberg and Zokai, 2000]. L'invariant de Fourier-Melin d'une image I est défini comme la transformée en coordonnées log-polaires du module de sa transformée de Fourier. Il est invariant en translation et une rotation ou un changement d'échelle de l'image se traduisent par des translations du FMI. Appliquer un SPOMF sur le FMI permet donc de recaler en rotation et en échelle deux images. Pour recaler en translation, il suffit d'appliquer les transformations de rotation et d'échelle sur l'une des deux images et d'appliquer un SPOMF.

Pour utiliser le FMI-SPOMF comme algorithme de reconnaissance, il est possible d'utiliser le PSNR du FMI-SPOMF, ou le PSNR du SPOMF appliqué entre le motif et l'image recalée en rotation et échelle.

Nous avons utilisé les formules suivantes pour la transformée en coordonnées log-polaires :

$$x(r,\theta) = \frac{W}{\sqrt{2}} K^{(r-R)} sin\left(\theta \frac{\pi}{T}\right)$$

$$y(r,\theta) = \frac{H}{2} + \frac{W}{\sqrt{2}} K^{(r-R)} cos\left(\theta \frac{\pi}{T}\right)$$

où (W,H) est la taille de l'image d'origine (la transformée de Fourier est donc de taille $\left(\frac{W}{2},H\right)$ après suppression de l'information redondante), (R,T) est la taille de l'image transformée et K un paramètre, qui détermine le plus grand facteur d'échelle que le FMI-SPOMF pourra recaler. Plus K est grand, moins le recalage en échelle sera précis. Nous avons utilisé $K = 1.03$, ce qui permet pour une image de 128 par 128 pixels un recalage d'un facteur 6. Les paramètres de rotation et d'échelle correspondant à une translation (x,y) sont donnés par :

$$echelle = K^x \;\; rotation = y\frac{\pi}{T}$$

La figure 4.8 donne des exemples de FMI-SPOMF, sur les mêmes images que la figure 4.7. On constate que le pic est beaucoup plus marqué et diminue plus rapidement avec l'écart d'orientation.

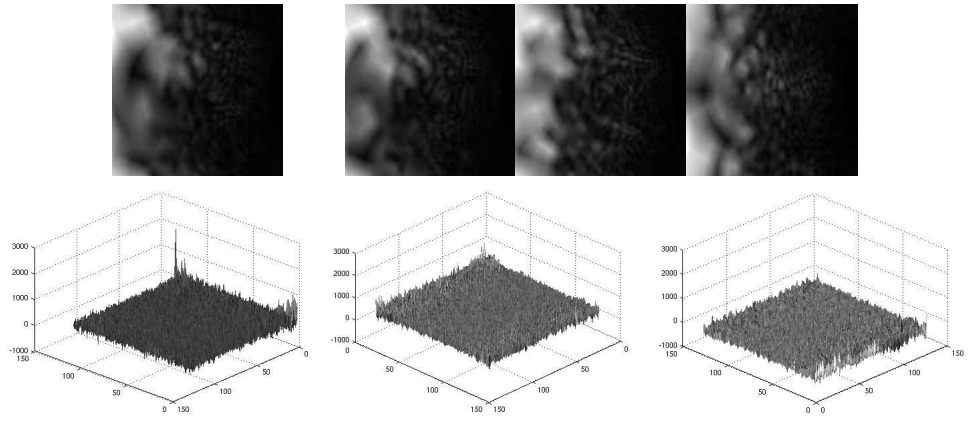


FIG. 4.8 – FMI pour l'image et les motifs d'angles -50, -70, 10 et FMI-SPOMF avec l'image.

De manière similaire au SPOMF, la transformée de Fourier doit être tronquée avant de calculer la différence de phase pour supprimer les composantes sans signification. Mais pour accélérer le calcul, les transformées de Fourier des FMI des motifs sont pré-calculées à l'initialisation. Nous avons donc décidé d'approximer la troncature correcte en coupant la transformée de Fourier du FMI pour $r > R + \frac{log\left(\frac{N}{W\sqrt{2}}\right)}{log(K)}$, ce qui correspond à conserver un cercle de rayon $N\sqrt{2}$ à la place d'un carré de côté N.

Pour éviter les artefacts provoqués par la troncature, un filtre de Hanning est appliqué sur toutes les images avant le calcul de leur transformée de Fourier. On calcule I' pour une image I de taille (w, h) :

$$\begin{aligned} I'(x,y) &= I(x,y)\left(0.5+0.5cos\left(\frac{2\,(x-w/2)\,\pi}{w}\right)\right) \\ &\quad \left(0.5+0.5cos\left(\frac{2\,(y-h/2)\,\pi}{h}\right)\right) \end{aligned}$$

Pré-traitements et autres améliorations. La qualité du recalage est un facteur important pour les performances des algorithmes de reconnaissance. Nous avons essayé d'appliquer divers filtres pour augmenter la stabilité de la forme de la tête sur les images dérivées. La forme elle-même étant plus stable que les intensités relatives des pixels la composant, nous avons tenté

d'employer des filtres d'égalisation d'histogramme et de seuillage. Les performances de ces filtres sont détaillées dans la partie résultats.

La présence d'un fond sur les images originales dégrade les performances des algorithmes de reconnaissance. Pour l'éliminer, il est possible d'utiliser les images dérivées pour générer un masque binaire, appliqué sur l'image avant la reconnaissance. Ce masque est créé en seuillant l'image dérivée et en appliquant une fermeture morphologique de la taille de l'objet, ce qui revient approximativement à prendre l'enveloppe convexe des points de l'objet supérieurs au seuil.

Résultats. Dans cette partie, nous détaillons d'abord le protocole et les critères d'évaluation des algorithmes de reconnaissance, ainsi que l'influence de leurs paramètres. Puis nous exposons brièvement pourquoi nous avons abandonné la corrélation croisée. Ensuite, nous exposons quelques résultats sur les performances de l'algorithme de recalage et son influence sur les performances globales. Enfin nous évaluons plus en détail les performances du SPOMF et du FMI-SPOMF.

Protocole expérimental. Pour pouvoir comparer les différents algorithmes, il faut que ceux-ci soient appliqués sur les même données. Pour ce faire, nous avons enregistré des séquences vidéos. Chaque séquence contient :

- 40 secondes d'images d'une tête de robot pour toutes les orientations dans [-80,80] par pas de 2 degrés, à 40 centimètres.
- 10 secondes d'images pour 64 orientations choisies aléatoirement dans [-80,80], à des distances de 40, 80 et 120 centimètres.

La première partie est utilisée pour l'apprentissage, La seconde pour les tests.

Un deuxième jeu de séquences a été enregistré, avec le *speaker* tourné de 180 degrés, afin d'obtenir des images pour tous les angles possibles.

Chaque séquence est enregistrée avec un fond différent et les tests sont effectués en utilisant une séquence différente pour l'apprentissage et les tests, afin d'être dans des conditions les plus proches possibles de l'utilisation de ces algorithmes.

Pour chaque test, nous mesurons les propriétés suivantes :

- Temps de calcul moyen par image.
- Pour différentes marges d'erreurs admissibles sur la valeur de l'orientation retournée (typiquement 5, 10 et 30 degrés) ;

- Taux de succès moyen : un test pour un angle donné est considéré comme ayant réussi si au bout des 10 secondes, la tête trouvée avec le meilleur score a une orientation correcte.
- Nombre moyen d'images nécessaires pour obtenir l'orientation correcte.
- Précision moyenne de la mesure de position et de distance.
- Taux de faux-positifs.

Ces indicateurs permettent de savoir si l'algorithme se comportera de manière satisfaisante lorsqu'il sera utilisée dans l'expérience des *Talking Robots*.

Évaluation de l'algorithme de corrélation croisée. Des tests préliminaires ont montré que l'algorithme de corrélation croisée ne convenait pas pour plusieurs raisons : le pic obtenu par la mesure de corrélation croisée est généralement très large [Lehmann, 1998]. Ainsi le score obtenu avec le motif d'orientation correcte, mais légèrement mal recalé est souvent très proche du score obtenu avec un mauvais motif. De plus, du fait de la nécessité d'appliquer la corrélation croisée en testant plusieurs valeurs de translation, cette méthode est plus lente d'un ordre de grandeur que les méthodes basées sur Fourier. Nous avons donc abandonné la corrélation croisée et nous nous concentrerons dans la suite sur le SPOMF et le FMI-SPOMF.

Évaluation des performances du recalage et de ses conséquences. Les performances de l'algorithme de recalage basé sur les moments semblent être assez mauvaises en pratique : de légères distorsions ou variations d'intensité de l'image provoquent des différences importantes sur les moments $M_{0,2}$ et $M_{2,0}$ et donc d'importantes erreurs de recalage en échelle. Mais l'impact de ces erreurs sur les performances globales est limité, pour deux raisons.

Tout d'abord, le SPOMF est assez résistant aux erreurs de recalage, comme le montre la figure 4, qui donne le logarithme du score obtenu avec le SPOMF entre une image de tête d'aibo orientée à -51° zoomée d'un facteur 0.8 1 et 1.2, et 33 motifs dans l'intervalle [-80,80]. Dans les trois cas, le résultat est correct à 20 degrés près.

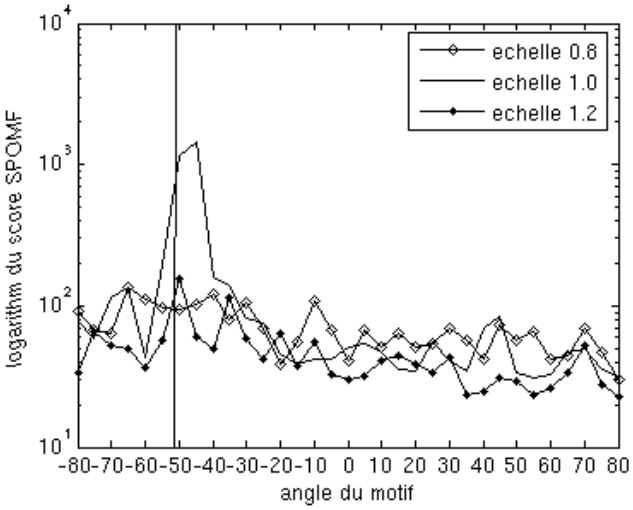


FIG. 4.10 – Score du SPOMF entre des motifs et un objet à différents zoom.

De plus, en cas d'erreur de recalage, le score donné par le SPOMF sera le même pour tous les motifs et d'une valeur plus faible qu'en cas de recalage correct. Comme le score est sommé d'image en image, le score final a de bonne chances d'être correct même si un pourcentage important des images sont mal recalées.

SPOMF. Comme dit précédemment, le SPOMF a été testé sur les images d'origine et les images dérivées. Nous présentons les résultats de ces deux tests séparément.

SPOMF sur les images dérivées. En nous limitant au domaine [-80, 80], avec un motif tous les 4° et un filtre de normalisation d'histogramme, nous obtenons les résultats suivants :

distance(cm)	40		80		120	
marge (degrés)	15	30	15	30	15	30
taux de succès	0.68	0.73	0.77	0.81	0.53	0.70

Supprimer le filtre dégrade les résultats d'environ 10%. Modifier le nombre d'images moyennées n'a qu'un effet marginal sur les résultats. L'erreur

moyenne sur la distance est de 10% et de 5 pixels sur la position, ce qui convient pour notre usage de l'algorithme.

Le temps moyen nécessaire pour trouver l'orientation correcte est de 14 images, soit environ deux secondes.

Le temps de calcul par image est de 110 millisecondes sur un pentium 4 cadencé à 2.8 GHz : l'algorithme peut être utilisé en temps réel.

Quand les tests sont effectués avec des motifs couvrant 360°, les résultats sont convenables à 40 centimètres, mais le taux de succès tombe en dessous de 50% pour toutes les distances supérieures : toutes les images d'orientation dans [-80, 80] sont systématiquement reconnues comme correspondant à des motifs du dos de la tête. Une explication possible est que les informations permettant de discriminer entre l'avant et l'arrière de la tête se trouvent dans des hautes fréquences qui ne sont pas disponibles quand l'objet est trop petit. Une étude plus approfondie serait nécessaire pour confirmer cette hypothèse.

SPOMF sur les images d'origine. Un phénomène similaire se produit lorsque le SPOMF est appliqué sur les images d'origine : les résultats sont convenables (0.92, 0.90, 0.70 pour une marge de 30° à respectivement 40, 80 et 120 centimètres) lorsqu'on se restreint à l'intervalle d'orientations [-80, 80], mais tombent en dessous de 50% sur 360°, très certainement pour la même raison.

FMI-SPOMF.

FMI-SPOMF sur les images dérivées. Le FMI-SPOMF appliqué sur des séquences d'images avec des orientations couvrant les 360° donne les résultats suivants :

distance (cm)	40		80		120	
marge (degrés)	15	30	15	30	15	30
taux de succès	1	1	0.75	0.93	0.48	0.65

Le taux de succès est trop bas pour une utilisation pratique. Une grande partie de l'information est perdue en ne considérant que les images dérivées. Comme elles ne contiennent que la forme de la tête, la plus grande partie de l'énergie du spectre de Fourier se trouve dans les hautes fréquences.

FMI-SPOMF sur les images d'origine. Le FMI-SPOMF appliqué sur les images originales avec des orientations couvrant les 360° donne les résultats suivants :

distance (cm)	40		80		120	
marge (degrés)	15	30	15	30	15	30
taux de succès	0.89	0.92	0.84	0.95	0.81	0.92

La mesure de distance est précise à 20% près et la mesure de position à 10 pixels près. Le temps de calcul est de 170 millisecondes par image à 40cm et diminue avec la taille de l'objet. L'intervalle entre les images étant de 160 millisecondes, environ 6% des images sont ignorées, ce qui a pour seul effet d'augmenter le temps de détection, qui est de l'ordre d'une seconde et demi en moyenne pour obtenir une orientation correcte.

Utiliser la technique de masquage décrite dans la partie 2.3.5 diminue le taux de succès de 20%. La principale raison est l'instabilité du masque qui coupe souvent des portions de l'objet.

Augmenter le nombre de motifs à un tous les 2° n'a pas d'effet notable sur le taux de succès. Le diminuer à un tous les 8° les diminue significativement : dans le pire des cas, l'objet et le motif le plus proche diffèrent d'une rotation de 4° (rotation dans l'espace et non dans le plan de l'image), ce qui provoque des différences suffisantes pour que le PSNR obtenu soit faible et de l'ordre de grandeur de celui obtenu avec les autres motifs d'orientation incorrecte. Une valeur de 4° entre les motifs semble être un bon compromis entre le temps de calcul et les performances.

4.2.4.2 Marquage des aibos.

Nous avons montré que le FMI-SPOMF pouvait être utilisé avec succès pour localiser une tête d'aibo et calculer sa distance et son orientation en temps-réel.

Néanmoins l'usage de cette méthode est assez contraignante : la position du corps et l'orientation verticale de la tête des deux robots doit être fixée, pour ne pas introduire de transformations de l'image non prises en compte dans le modèle. De plus, le mouvement d'oscillation constant de la tête risque de diminuer significativement la durée de vie des moteurs de l'aibo.

Le problème est considérablement simplifié si l'on s'autorise à modifier l'aibo pour faciliter sa détection. La solution la plus simple consiste à coller des morceaux de cartons de couleur vive sur sa surface. L'approche classique pour les détecter consiste à détecter dans l'image tous les pixels dont la valeur en YCrCb est dans un intervalle tridimensionnel fixé, à appliquer une fermeture morphologique et à détecter les composantes connexes. La couleur perçue par la caméra étant très dépendante des conditions d'éclairages, des intervalles très larges doivent être utilisés, ce qui donne lieu à de nombreuses fausses détections. Pour rendre le processus plus robuste, nous utilisons donc des patches de deux couleurs à proximité.

Nous avons choisis les couleurs magenta et cyan, expérimentalement les plus faciles à détecter. Nous avons placé des patches carrés mi magenta mi cyan à l'avant, à l'arrière et sur les côtés des aibos, en orientant chaque patch de manière différente. Nous avons aussi placé quatre patches circulaires, constitués de deux anneaux concentriques de magenta et cyan au-dessus de chaque patte.

L'algorithme de détection procède comme suit : tout d'abord, tous les patches de couleur magenta et cyan sont listés. Puis le système teste toutes les combinaisons de deux patches de couleur différentes et compare leurs centres de gravité, surfaces et boîtes englobantes :

Si la différence relative des aires est inférieure à 1.4 $\left(\frac{|a_1-a_2|}{\frac{1}{2}(a_1+a_2)}\right)$, la différence relative des coordonnées y inférieure à 0.6 et de la variance sur y inférieure à 0.75 et si la distance sur l'axe x des frontières des boîtes englobantes divisée par la largeur moyenne est inférieure à 0.3, alors ces deux patches sont considérés comme étant l'un au-dessus de l'autre. Le même algorithme en inversant les coordonnées x et y détecte les patches l'un à côté de l'autre, et un algorithme similaire détecte les patches concentriques. Cette procédure peut sembler très *ad hoc*, mais il s'agit simplement de la définition mathématique de "deux rectangles de même taille l'un au-dessus de l'autre". Le grand nombre de critères permet d'utiliser des seuils de tolérance très élevés sans provoquer de faux positifs s'ils sont tous satisfaits.

Pour obtenir l'orientation à partir des patches, l'algorithme se base sur l'information de visibilité de chaque patch carré et la taille et la position des patchs circulaire relative aux patchs carrés.

Cette approche permet d'obtenir une information de distance peu précise et une orientation à 30 degrés près, mais avec très peu de faux positifs. Cette précision nous convient. La synchronisation précise des champs de vision est

effectuée par pointage laser comme exposé plus haut.

4.3 Mécanismes fondamentaux

4.3.1 Référents

Les agents des *Talking Heads* associent des symboles à des propriétés des objets de leurs perceptions, qui sont les référents de ces symboles. Ce modèle nécessite qu'un concept d'objet puisse être défini et instancié à partir de la seule perception des agents. L'environnement des *Talking Heads* est un modèle simplifié du monde possédant cette propriété : il est constitué de formes géométriques de couleur placées sur un tableau blanc. L'image d'une partie de cet environnement perçue par un agent peut être facilement segmentée, pour donner directement un ensemble d'objets associés à leurs propriétés.

Nous avons choisi de conserver ce paradigme simplifié pour cette première version des *Talking Robots* et ce malgré l'environnement non contraint qui n'offre pas *a priori* d'objets stables s'extrayant trivialement d'une image : un objet est défini par une région d'une image, donnée par un algorithme de segmentation.

Dans la section suivante, nous exposons le protocole expérimental que nous avons mis en place pour choisir l'algorithme de segmentation le plus adapté à notre problème parmi l'existant et les résultats obtenus. Ces travaux peuvent être retrouvés dans [Baillie and Nottale, 2005b].

Choix de l'algorithme de segmentation. Afin de sélectionner l'algorithme de segmentation le plus approprié à notre tâche, il nous faut tout d'abord définir un moyen d'évaluer les performances d'un algorithme de segmentation.

Les techniques classiques consistent principalement à tester l'algorithme sur une base d'images pour laquelle une vérité terrain est disponible et à mesurer une erreur entre la segmentation et la vérité terrain [Huang and Dom, 1995][Correia and Pereira, 2003].

Cette approche ne nous convient pas. Notre principal critère de choix est un critère de robustesse : pour notre expérience, deux segmentations de la même scène de deux points de vue légèrement différents doivent être très proches. Nous faisons ici l'hypothèse que ce critère est équivalent à la

robustesse de l'algorithme à un bruit sur l'image segmentée afin de pouvoir l'évaluer plus simplement.

Nous présentons deux mesures de distance entre deux segmentations que nous avons développées. Puis nous utilisons ces mesures pour comparer la stabilité de différents algorithmes.

Outils. Étant données deux segmentations $A = \{A_1, ..., A_n\}$ et $B = \{B_1, ..., B_m\}$, définies comme un ensemble d'ensembles disjoints de pixels, définissons la matrice M_{AB} et les matrices normalisées $M_{A/B}$ et $M_{B/A}$:

$$M_{i,j}^{AB} = card\left(A_i \cap B_j\right)$$

$$M_{i,j}^{A/B} = \frac{M_{i,j}^{AB}}{card\left(A_i\right)},\ M_{i,j}^{B/A} = \frac{M_{i,j}^{AB}}{card\left(B_j\right)}$$

La valeur (i, j) de $M_{A/B}$ représente la proportion de la région A_i intersectant B_j. La somme des valeurs sur chaque ligne est donc 1. Si les deux segmentations sont très proches, la matrice aura une valeur élevée par ligne et par colonne. Sinon, la variance par ligne/colonne sera moins importante.

Mesure entropique. Chaque ligne des matrices $M_{A/B}$ et $M_{B/A}$ peut être considérée comme une probabilité de distribution. La mesure d'entropie peut donner une bonne estimation de la dispersion des valeurs :

$$E_i(M) = \sum_j -M_{i,j} log\left(M_{i,j}\right)$$

Notre première mesure de stabilité est donc définie comme la moyenne de l'entropie de chaque ligne.

$$S_{ent}\left(A, B\right) = 1 - max\left(\frac{1}{n}\sum_i E_i\left(M^{A/B}\right), \frac{1}{m}\sum_i E_i\left(M^{B/A}\right)\right)$$

Le max permet de rendre cette mesure symétrique. Les expériences montrent que les mesures d'entropie moyenne peuvent être très différentes entre $M^{A/B}$ et $M^{B/A}$, par exemple si B est une sur-segmentation de A.

Mesure de correspondance Définissons $C^{A/B}(i)$ comme l'index de la région de B *correspondant* à la région d'index i dans A, par :

$$C^{A/B}(i) = argmax_j \left(M_{i,j}^{A/B} \right)$$

$C^{A/B}(i)$ est la région intersectant le plus la région i. $C^{B/A}$ est défini de manière similaire. Nous pouvons alors définir la proportion des points de A qui sont dans leur région correspondante dans B par :

$$R^{A/B} = \frac{\sum_i M_{i,C^{A/B}(i)}^{AB}}{\sum_i card(A_i)}$$

ce qui donne la mesure de stabilité :

$$S_{corr} = min \left(R^{A/B}, R^{B/A} \right)$$

$\sum_i card(A_i)$ est l'aire (en pixels) de l'image si la segmentation la recouvre entièrement.

Comparaison des deux mesures. Ces deux mesures ne sont pas équivalentes, comme le montre l'exemple trivial de la figure 4.11 : a est l'image de référence. Dans b, un des carrés est translaté de 30 pixels ; dans c les deux carrés sont translatés de 15 pixels. La mesure S_{corr} est la même dans les deux cas (0.91), mais la mesure S_{ent} diffère (0.70 pour a-b et 0.54 pour a-c) : un S_{ent} haut signifie que l'erreur est concentrée dans quelques régions et un S_{ent} bas qu'elle est plus distribuée.

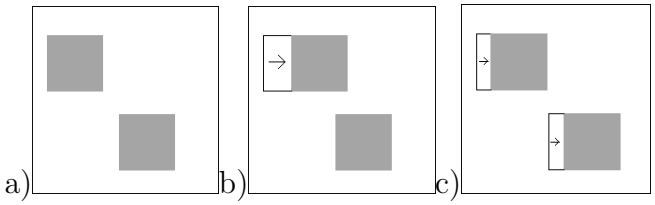


FIG. 4.11 – Illustration des propriétés de S_{ent} et S_{corr}

Utilisation des mesures pour comparer les algorithmes. Nous avons sélectionné quatre algorithmes de segmentation : l'algorithme de croissance de région, RHS (*recursive histogram splitting*), CSC (*color structure code*)

[Priese and Rehrmann, 1993] et l'algorithme FSA (*feature-space analysis*) décrit dans [Comaniciu and Meer, 1997] (d'autres ont été considérés, mais les résultats de l'évaluation ne sont pas rapportés ici). Pour mesurer l'impact du bruit sur ces algorithmes, nous avons sélectionné 3 images de complexité différentes : Lena, Pepper et une image prise dans le laboratoire avec la caméra de l'aibo. Nous avons appliqué différents niveaux de bruit gaussien (la modélisation du bruit sur un flux vidéo la plus simple [Boie and Cox, 1992]) à ces images, puis appliqué nos mesures à toutes les paires d'images obtenues, en ignorant les régions de moins de 10 pixels de coté qui sont principalement des artefacts dus au bruit.

Comme les mesures de stabilité diminuent quand le nombre de régions augmente, les paramètres des algorithmes de segmentation ont été réglés de manière à obtenir approximativement le même nombre de régions pour chacun. Ce nombre de régions a été choisi en fonction du nombre d'objets visibles par un observateur humain, entre 20 et 40.

La figure 4.12 résume les résultats obtenus:

Les algorithmes CSC et FSA sont les plus stables d'après nos deux critères pour de faibles valeurs de bruit. CSC est significativement meilleur pour S_{coor} et moins bon sur le critère d'entropie pour des bruits plus forts. Les graphes de CSC sont similaires pour les trois images, ses performances dépendent peu de la nature de l'image, alors que le FSA exhibe des courbes très différentes d'une image à l'autre.

Pour valider notre modèle, nous avons utilisé nos mesures sur une série d'images de la scène c), prises avec un capteur CMOS de faible résolution à intervalle d'une seconde. La mesure moyenne obtenue après comparaison de toutes les paires de segmentations est donnée dans le tableau suivant :

Algo.	$S_{entropy}$	S_{corr}
FSA	0.81	0.87
CSC	0.82	0.93
RG	0.66	0.85
RHS	0.63	0.76

La variance de l'histogramme des différences entre toutes les paires d'images nous permet d'estimer un niveau de bruit de 2.5. Les valeurs de $S_{entropy}$ correspondent à ce que nous avons obtenu pour l'image c à ce niveau de bruit, et les valeurs de S_{corr} sont plus basses, mais de manière cohérente.

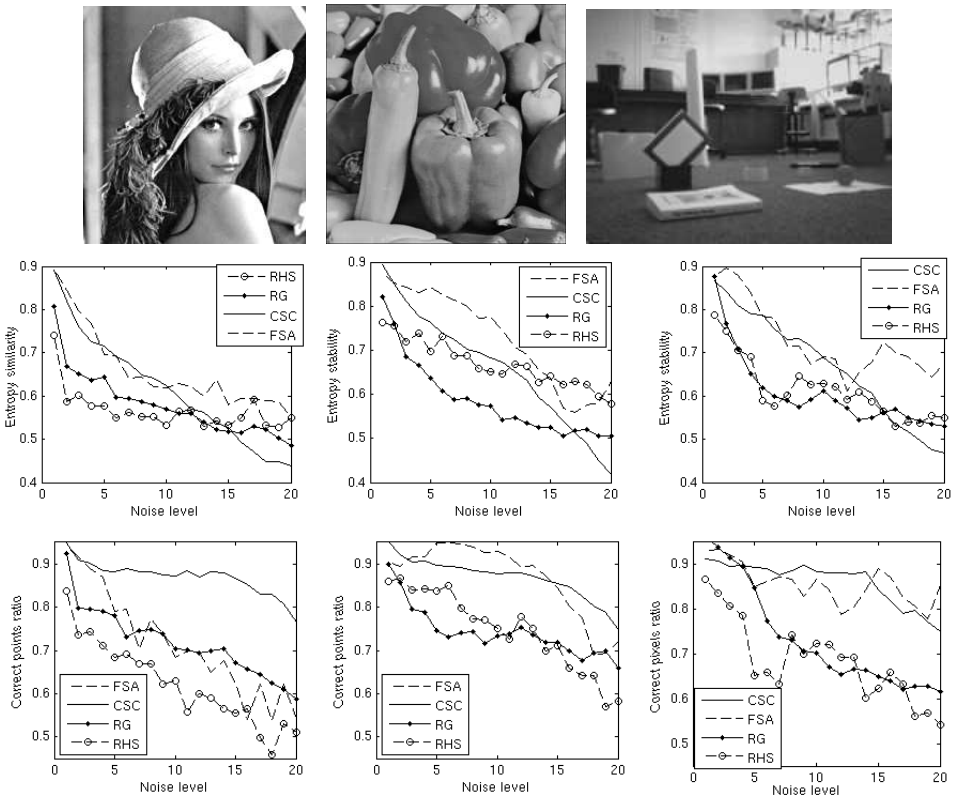


FIG. 4.12 – Stabilité pour les images Lena, peppers et une image de notre laboratoire

La cohérence des résultats des deux mesures entre ces expériences est un bon indicateur de la capacité de ces mesures à évaluer la stabilité d'algorithmes de segmentation.

Agrégation de multiples segmentations. Nous venons de montrer que des mesures de stabilité pouvaient être utilisées pour évaluer des algorithmes de segmentation, mais il est possible d'aller plus loin : si un flux d'image est disponible, il est possible d'agréger les segmentations effectuées sur chaque image pour obtenir une segmentation encore plus robuste. Nous introduisons dans la section suivante deux algorithmes d'agrégation, l'un basé sur les frontières des régions utilisant la morphologie mathématique, l'autre basé sur les régions et une approche ensembliste, puis nous comparons leurs performances.

Agrégation basée sur les contours des régions. Cet algorithme inspiré de [Manzanera, 2002] se base sur la considération suivante : les frontières entre régions qui se trouvent sur toutes les segmentations doivent être gardées et celles qui se trouvent uniquement sur une petite fraction des segmentations doivent être éliminées.

Chacune des segmentations est transformée en une image de contour binaire : un pixel est marqué 1 s'il est 8-connexe à au moins deux régions différentes, 0 sinon. Toutes les cartes de contour sont ajoutées et une dilatation avec pour élément structurant un cercle de 2 pixels est effectuée, afin de fusionner les frontières proches. Le résultat est seuillé, le seuil S étant compris entre 1 et le nombre de segmentations et déterminant la proportion des segmentations devant contenir une frontière pour que celle-ci soit gardée. Finalement l'image obtenue est convertie à nouveau en région en détectant les composantes connexes.

Agrégation basée sur les régions. Cet algorithme utilise le concept de correspondance entre régions de deux segmentations exposé plus haut. Le résultat n'est qu'une segmentation partielle, certains pixels ne font partie d'aucune région.

La segmentation agrégée est initialisée avec la première segmentation. Les suivantes sont agrégées séquentiellement en utilisant l'algorithme suivant : pour chaque pixel, soient i et j les régions de la segmentation agrégée et de la segmentation à intégrer auxquelles ce pixel appartient. Le pixel est maintenu dans la région i si les régions i et j sont des régions qui se correspondent : $C^{A/B}(i) = j$ et $C^{B/A}(j) = i$. Sinon, le pixel est enlevé de toute région.

L'effet de cet algorithme est de réduire chaque région aux pixels appartenant à cette région pour toutes les segmentations.

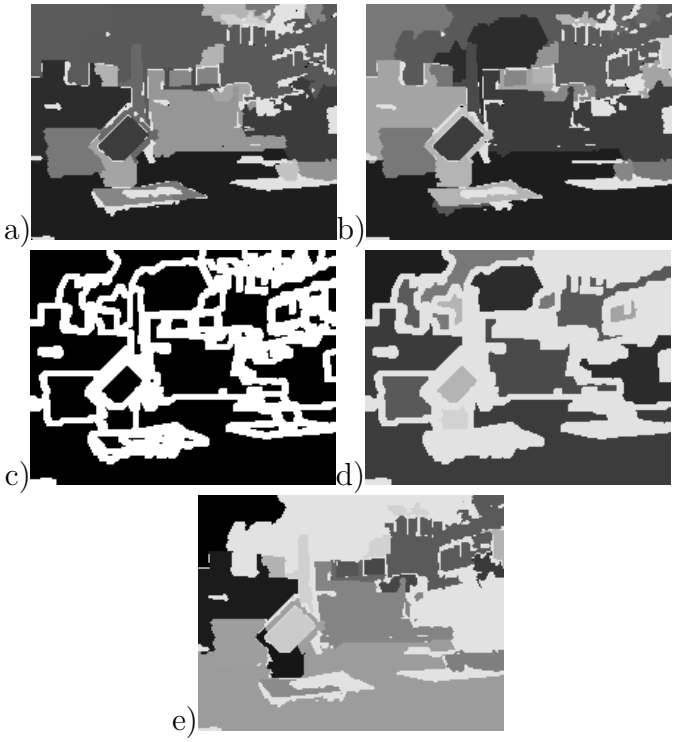


FIG. 4.13 – Exemple d'agrégation pour les deux algorithmes a) et b): segmentations initiales. c) countours agrégés d) segmentation résultante e) agrégation par région.

Performance des algorithmes d'agrégation. Pour tester ces deux algorithmes d'agrégation, nous avons enregistré 100 images de la scène c) à intervalle d'une seconde. Nous avons appliqué chaque algorithme d'agrégation sur des groupes de 5 et 10 segmentations d'images successives, pour les quatre algorithmes de segmentation. La procédure d'évaluation est la même que celle utilisée précédemment. Le tableau suivant compare les scores moyens obtenus pour des agrégations de 5 images. Pour l'agrégation par contours, le seuil optimal déterminé expérimentalement a été utilisé dans chaque cas.

Agreg:	sans agrégation		agrégation par contour		agrégation par régions	
	S_{corr}	S_{ent}	S_{coor}	S_{ent}	S_{corr}	S_{ent}
FSA	0.91	0.85	0.93	0.91	0.84	0.91
CSC	0.89	0.80	0.92	0.94	0.91	0.95
RG	0.91	0.69	0.82	0.91	0.78	0.88
RHS	0.76	0.63	0.80	0.89	0.80	0.90

Les deux algorithmes d'agrégation augmentent la stabilité des quatre algorithmes de segmentation sur le critère S_{ent}. Les résultats pour S_{corr} sont moins clairs : l'agrégation par contours a un effet positif sur CSC, FSA et RHS et très négatif sur RG tandis que l'agrégation par région améliore légèrement CSC et RHS et dégrade les deux autres. Augmenter le nombre de segmentations agrégées à 10 améliore légèrement les résultats de CSC et n'a pas d'impact sur les autres algorithmes.

On note aussi que les meilleurs résultats de l'agrégation par contours sont obtenus avec un seuil de 1 pour CSC et un seuil élevé pour FSA, ce seuil n'ayant pas d'impact sur RG et RHS.

Les effets des deux algorithmes d'agrégation sont donc très dépendants de l'algorithme de segmentation utilisé et probablement du type de l'image segmentée. Néanmoins l'amélioration observée pour CSC est cohérente et significative.

Conclusions. Au vu de ces résultats, nous avons choisi de conserver CSC et l'algorithme d'agrégation par régions. Les régions de moins de 10 pixels de côté sont filtrées. Les paramètres de CSC sont réglés de manière à ce qu'une image typique contienne de 5 à 20 régions segmentées.

4.3.2 Signifiants et canaux perceptifs

De manière similaire aux *Talking Heads*, chaque objet est perçu par un *Talking Robot* comme un ensemble de valeurs entre 0 et 1, correspondant chacune à un canal visuel. Ces canaux correspondent à des propriétés visuelles de ces objets. Nous avons utilisé dans nos expériences les canaux suivants :

- position horizontale et verticale (HPSO, VPOS)
- largeur, hauteur, aire (WIDTH, HEIGHT, AREA)
- couleurs dans différents espaces (GRAY,R, G, B, HUE, SAT, VAL)

– élongation : rapport hauteur/largeur (ELONG)

Ces valeurs sont normalisées par le contexte : pour chaque canal, l'objet à la valeur la plus basse se retrouve avec la valeur 0 et l'objet avec la valeur la plus haute se retrouve avec la valeur 1.

4.3.3 Arbres de discrimination

Ces valeurs sont catégorisées par des arbres de discrimination, identiques aux *discrimination trees* des *Talking Heads*. (cf. 2.4 pour une description détaillée). Un arbre différent est associé à chaque canal visuel. Le taux de succès utilisé pour décider d'élaguer ou de faire croître un arbre est donné par la formule itérative suivante :$s_t = s_{t-1} * decay + \Delta_{success} * (1 - decay)$, $\Delta_{success}$valant 1 si l'itération t du jeux de langage en cours est un succès, 0 si c'est un échec. *decay* est une constante fixant la réactivité du système fixée à 0.9 dans nos expérimentations. À chaque itération, l'élagage d'un nœud est déclenché avec une probabilité de $P_e.s_t$ et la croissance d'un nœud avec une probabilité de $P_c.(1 - s_t)$, P_e et P_c étant deux constantes déterminant le taux de succès cible : en effet, la taille des arbres de discrimination se stabilise si les probabilités de croissance et d'élagage sont égales, soit pour $s_t = \frac{P_c}{P_c+P_e}$. Si s_t est en dessous de cette valeur, la taille des arbres augmente, facilitant la discrimination entre objets et augmentant le taux de succès. Si s_t est au dessus, les arbres s'élaguent, diminuant le taux de succès. Les valeurs $P_e = 0.04$ et $P_c = 0.4$ sont utilisées dans nos expériences, ce qui donne un taux de succès à arbres stables de 0.9.

Observons que si ce taux de succès n'est jamais atteint, les arbres continueront de croître indéfiniment.

Le nœud à croître est choisi aléatoirement parmi les 20% des feuilles les plus utilisées et le nœud à élaguer parmi les 20% des nœuds avec le score le plus faible. L'utilisation d'un nœud est défini comme le nombre total d'objets perçus ayant leur propriété associée à ce nœud. Son score est défini comme le pourcentage des itérations du jeu de langage faisant intervenir ce nœud s'étant soldées par un succès (ce score n'est donc défini que pour le *guessing game*).

4.3.4 Symbole et combinaison de nœuds

Les symboles utilisés par les *Talking Robots* et auxquels sont associés les mots du lexique sont les nœuds des arbres de discrimination (par exemple,

HPOS[0,0.5], ou GREY[0.625, 0.75]). Nous avons envisagé d'autoriser aussi les associations entre un symbole et une combinaisons de plusieurs nœuds. Le principal avantage est que la discrimination entre les objets est facilitée car elle peut se faire avec des arbres de discrimination moins profonds : si un robot peut utiliser deux canaux au lieu d'un pour discriminer un objet d'un contexte, il pourra utiliser des intervalles de valeurs plus larges sur chacun de ces canaux. Les robots utilisant des classifications plus grossières, celles-ci ont plus de chances d'être partagées, car elles seront moins sensibles aux différences de perceptions entre eux.

Néanmoins, ce choix s'est avéré inefficace : il complique grandement la tâche d'apprentissage du *hearer*. En effet, lorsque celui-ci doit apprendre une association entre un mot et un signifiant, le nombre de signifiants possible est considérablement augmenté, car toutes les combinaisons de deux nœuds discriminant l'objet des autres sont des candidats possibles. On observe bien une diminution de la profondeur des arbres lors du *discrimination game*, mais les lexiques des agents ne convergent absolument pas.

4.3.5 Mots, lexique

Les mots utilisés par les agents sont des combinaisons de deux bi-syllabes. Les bi-syllabes sont les 144 combinaisons des syllabes "ka ke ki ko ma me mi mo sa se si so". Ceux-ci sont échangés sous formes de chaînes de caractères par le réseau. Le lexique est identique à celui des *Talking Heads*. Il est composé pour chaque agent d'associations pondérées entre un mot et un signifiant, soit un nœud d'un de ses arbres de discrimination. Le poids de chaque entrée est augmenté lorsqu'elle est utilisée avec succès dans une itération du *guessing game*. Ce poids est diminué dans trois cas : quand l'entrée est utilisée dans une itération du *guessing game* qui échoue, lorsque le mot est le même que celui d'une autre entrée reconnue avec succès par le *hearer* (homonymie) et lorsque le signifiant est le même que celui d'une autre entrée utilisée avec succès par le *speaker* (synonymie).

4.4 Jeu de discrimination

Le jeu de discrimination, mené par un agent seul, est identique à sa version des *Talking Heads* : un agent seul se déplace dans l'environnement, choisit aléatoirement un angle de rotation de la tête et segmente l'image acquise

par sa caméra. Il choisit aléatoirement un des objets obtenus et cherche un symbole (nœud des arbres de discriminations) qui ne peut se référer qu'à cet objet et aucun autre du contexte. S'il trouve un tel symbole, le jeu est un succès, sinon le jeu est un échec. Le taux de succès est mis à jour et les arbres de discrimination sont adaptés en fonction des règles exposées dans les sections précédentes. Si tout se passe bien, les arbres de discrimination se stabilisent, avec un taux de succès élevé. Sinon, leur taille croît indéfiniment et le taux de succès reste faible.

4.5 *Guessing Game*

De même, le *guessing game* est similaire dans son principe à sa version des *Talking Heads*.

Les deux robots se positionnent côte à côte. Le *speaker* pointe aléatoirement au laser un point de son environnement, puis les deux robots centrent leurs champs de vision sur le spot laser. Chaque robot segmente l'image de sa caméra pour obtenir une dizaine d'objets.

A chaque objet est associé une valeur normalisée par canal perceptif et les saliences sont calculées, comme indiqué en 4.3.2. Le *speaker* choisit aléatoirement un sujet parmi les objets dont la salience est supérieure à un seuil. Puis il classe les signifiants discriminant par salience et choisit le premier ayant un mot associé dans son lexique. S'il ne trouve aucun signifiant, l'itération est un échec. S'il ne trouve aucun mot, un nouveau mot est créé aléatoirement. Le *speaker* communique le mot au *hearer*.

Le *hearer* reçoit le mot. Il classe par leur poids toutes les entrées de son lexique utilisant ce mot et choisit la première dont le signifiant a exactement un objet référent dans son contexte. Il pointe alors cet objet au *speaker*. S'il ne trouve aucun objet, il le signale au *speaker*.

Le *speaker* vérifie si le *hearer* pointe bien le sujet. Si oui, l'itération est un succès. Si non, ou si le *hearer* ne pointe rien, le *speaker* pointe le sujet au *hearer*. L'itération est alors un échec. Dans ce cas, le *hearer* apprend une nouvelle association entre le mot entendu et le signifiant le plus salient discriminant l'objet pointé par le *speaker*.

A l'issue de chaque itération, les arbres de discrimination sont élagués ou grossis selon la procédure exposée dans les sections précédentes et les pondérations des entrées du lexique sont mises à jour.

4.6 Implantation

Cette première version des *Talking Robots* a été implantée en C++ et utilise le langage URBI à travers la liburbi pour communiquer avec les aibos. Une bibliothèque C++ a été développée pour pouvoir exprimer simplement et de manière modulaire des graphes de comportement. Le lecteur intéressé pourra se reporter à l'annexe A pour de plus amples détails.

4.7 Résultats

Jeu de discrimination. Afin de démarrer le *guessing game* avec des arbres de discrimination suffisamment fournis, chaque agent effectue quelques centaines d'itérations du jeu de discrimination seul, qui correspond au début de la partie *speaker* du *guessing game*, jusqu'au choix du sujet. On observe (fig. 4.14)qu'avec les canaux hauteur, largeur, position X,Y, couleur en RGB, élongation et aire, le nombre de nœuds augmente jusqu'à se stabiliser, de même que le taux de succès.

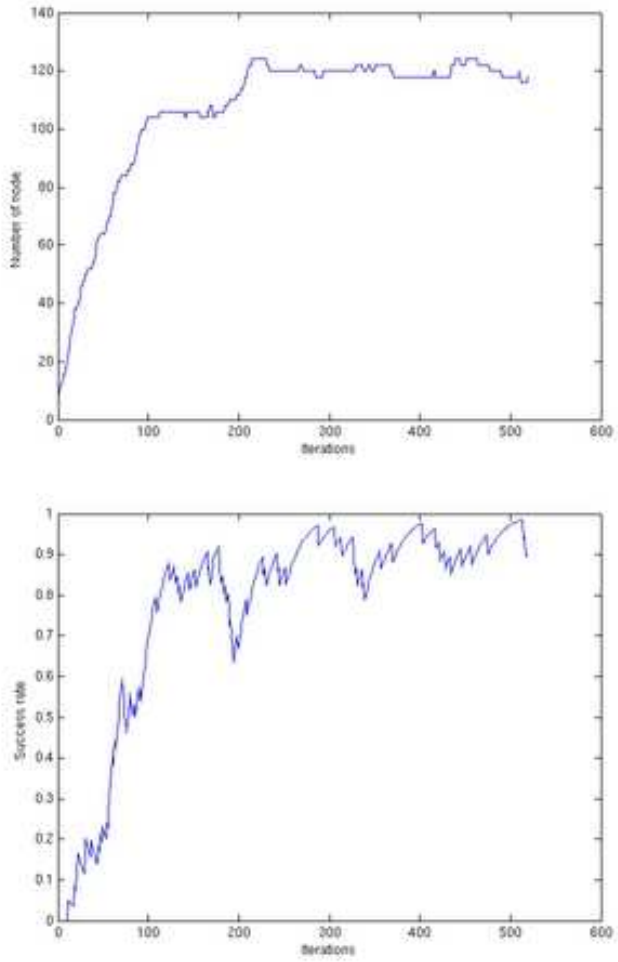


FIG. 4.14 – Évolution du taux de succès et du nombre de nœuds d'un *discrimination game*

Afin de vérifier la propriété d'adaptabilité, nous avons artificiellement forcé la valeur du canal saturation à 0 sur tous les objets. En quelques dizaines d'itérations, on constate (figure 4.15) que l'arbre de discrimination de ce canal s'atrophie complètement.

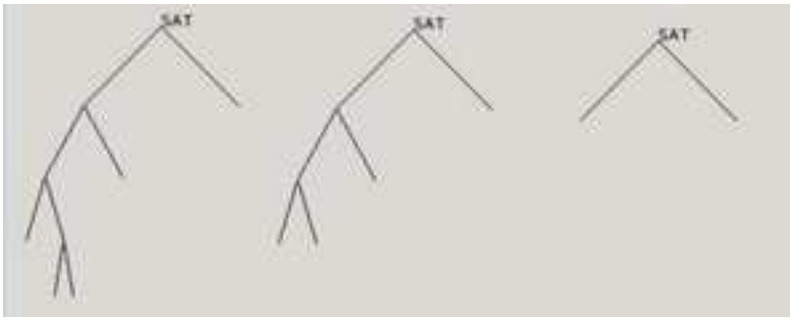


FIG. 4.15 – Adaptation du canal Saturation à l'absence d'information pertinente

Néanmoins, le succès du jeu de discrimination ne présage en rien de la robustesse des catégories utilisées par les agents et donc du succès du *guessing game*. En effet, le fait qu'un agent soit capable de discriminer les objets de son environnement n'implique pas que les catégories qu'il utilise soient partagées par un autre agent ayant une perception légèrement différente de la scène. Formulé autrement, si on remplace les perceptions d'un agent par des valeurs tirées au hasard, le jeu de discrimination se passera de la même manière.

Guessing game. Même avec uniquement deux agents alternant le rôle de *speaker* et *hearer*, nous n'avons pas réussi à obtenir convergence des lexiques entre les robots. La figure 4.16 montre un exemple typique d'évolution du taux de succès, qui ne dépasse pas 0.5 dans le meilleur des cas.

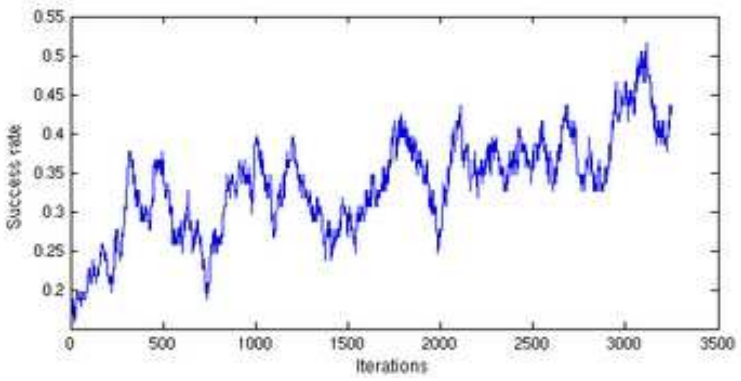
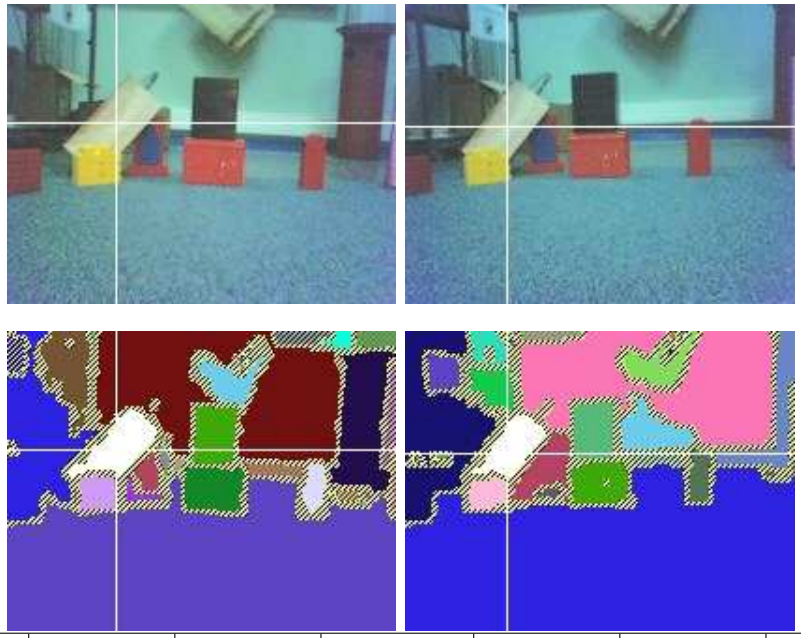


FIG. 4.16 – Exemple typique d'évolution du taux de succès d'un *guessing game*.

Nous avons essayé de minimiser les erreurs en modifiant certains éléments :

- Nous avons introduit des canaux *a priori* plus robustes au bruit comme l'orientation ou l'élongation.
- Nous avons expérimenté en confiant toujours le rôle de *speaker* au même robot, ce qui signifie qu'il est le seul à créer des mots, suivant le modèle d'apprentissage *Iterated Learning Mode* [Vogt and Coumans, 2003], mais cela a peu d'impact sur le taux de succès.

La cause plus fondamentale du problème est que plus de la moitié des associations apprises par le *hearer* sont "fausses" : le sens associé au mot n'est pas celui choisi par le *speaker*. Les saliences sont en effet très basses du fait du nombre relativement important d'objets et les différences entre les perceptions des deux agents (petits écarts sur les limites des régions, voire fusion de deux régions pour un des agents) sont suffisantes pour faire varier les valeurs des canaux du même ordre de grandeur que ces saliences. En conséquence, le canal le plus salient pour un objet n'a que peu de chances d'être le même pour les deux robots, même si cet objet est perçu de la même manière par les deux agents : le contexte sera différent.



CR	CB	AREA	HPOS	VPOS	WIDTH	HEIGHT
0.4821(0.013)	0.4420(0.008)	0.0228(0.001)	0.2837(0.052)	0.3750(0.012)	0.2260(0.004)	0.2188(0.031)
0.5045(0.008)	0.4955(0.000)	0.0690(0.020)	0.0721(0.081)	0.3625(0.012)	0.2308(0.004)	0.6250(0.112)
0.4062(0.000)	0.5580(0.004)	0.4084(0.242)	0.5048(0.019)	0.7875(0.250)	0.9952(0.375)	0.5125(0.043)
0.4062(0.000)	0.5179(0.004)	0.0236(0.001)	0.1538(0.076)	0.1187(0.043)	0.1250(0.014)	0.3375(0.093)
0.5670(0.031)	0.2321(0.192)	0.0100(0.003)	0.2308(0.052)	0.5312(0.006)	0.0962(0.000)	0.1125(0.012)
0.3884(0.017)	0.5045(0.008)	0.1665(0.097)	0.5529(0.014)	0.1937(0.031)	0.6202(0.375)	0.4313(0.037)
0.5982(0.031)	0.4598(0.008)	0.0028(0.001)	0.3365(0.024)	0.5375(0.006)	0.0962(0.000)	0.0812(0.025)
0.4688(0.013)	0.5625(0.004)	0.0053(0.002)	0.3606(0.024)	0.4688(0.043)	0.0577(0.000)	0.1250(0.012)
0.7009(0.044)	0.4241(0.017)	0.0200(0.001)	0.5240(0.004)	0.5188(0.006)	0.1490(0.019)	0.1437(0.018)
0.4955(0.008)	0.4955(0.000)	0.0194(0.001)	0.5288(0.004)	0.3438(0.018)	0.1106(0.014)	0.1875(0.012)
0.4509(0.017)	0.4821(0.013)	0.0130(0.003)	0.5673(0.014)	0.1625(0.031)	0.1683(0.000)	0.1750(0.012)
0.6562(0.044)	0.4509(0.008)	0.0074(0.002)	0.7885(0.067)	0.5125(0.006)	0.0577(0.000)	0.1625(0.012)
0.4152(0.008)	0.5446(0.013)	0.0029(0.001)	0.8558(0.052)	0.0250(0.093)	0.0721(0.014)	0.0562(0.025)
0.5179(0.013)	0.5134(0.004)	0.0491(0.020)	0.9087(0.052)	0.2937(0.050)	0.1683(0.000)	0.4688(0.037)

FIG. 4.17 – Exemple de deux segmentations par les deux agents et table des valeurs sur les différents canaux pour l'un des deux. A chaque ligne correspond un objet. Le sujet en blanc sur les segmentations est la première ligne du tableau, les saliences soulignées sont les plus hautes saliences par objet.

Filtrer les objets en fonction de la salience pour ne garder que ceux au-dessus d'un certain seuil sur au moins un canal comme le font les *Talking Heads* pose un problème supplémentaire : très fréquemment certains objets sont filtrés par un agent et non par l'autre, ce qui modifie complètement la répartition des saliences.

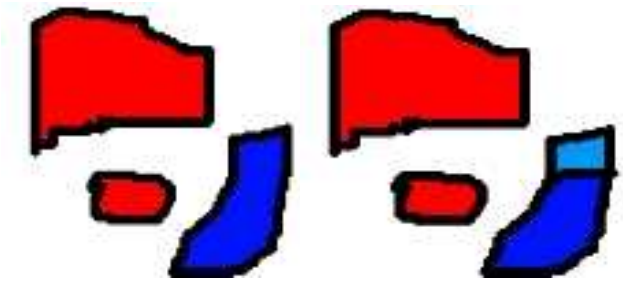


FIG. 4.18 – Exemple trivial montrant l'effet des écarts de segmentation sur la salience : le sujet est l'objet central et les canaux considérés HPOS VPOS WIDTH HEIGHT AREA et COLOR. Sur la première image le canal salient est sa hauteur et sur la deuxième sa position verticale, alors que le sujet est perçu à l'identique par les deux agents.

4.8 Analyse

Techniquement, notre première version des *Talking Robots* ne fonctionne pas car les écarts de perception entre les deux agents sont du même ordre de grandeur que les saliences de ces perceptions : Le point-clef du mécanisme d'apprentissage des *Talking Heads* est le moment où le *hearer* apprend une association entre le mot prononcé par le *speaker* et un des signifiants discriminant l'objet pointé par le *speaker*. Celui-ci doit trouver le même signifiant que le *speaker* pour apprendre correctement. Il choisit un signifiant portant sur le canal visuel le plus salient, ce qui est le bon choix si deux conditions sont respectées :

- Le *speaker* a initialement choisi la propriété la plus saliente de l'objet lorsqu'il a cherché un signifiant puis un mot s'y référant dans son lexique. Cette condition est toujours vérifiée, car ce choix est codé en dur dans le comportement du *speaker*.
- La propriété la plus saliente du sujet est la même pour les deux agents.

Cette deuxième condition n'est vraie que si les perceptions des deux agents sont suffisamment proches et dépend de la perception de tous les objets du contexte et non uniquement du sujet. Si les perceptions diffèrent trop, la propriété la plus saliente du sujet ne sera pas la même pour les deux agents et l'association mot-signifiant apprise par le *hearer* ne sera pas la bonne.

Plus fondamentalement, l'expérience des *Talking Heads* utilise comme premier niveau de perception une information déjà très abstraite : un ensemble d'objets, associé chacun à des propriétés visuelles. Les expériences décrites dans ce chapitre tendent à montrer qu'un algorithme de segmentation fonctionnant sur une image seule, sans *a priori*, ne peut fournir une telle abstraction de manière suffisamment robuste.

Chapitre 5

Talking Robots 2.0, modèle d'objet

5.1 Présentation

Partant du constat établi au chapitre précédent, nous avons entrepris la conception d'un nouveau système de perception visuelle pour les *Talking Robots*, inspiré de l'état de l'art en détection et reconnaissance d'objets, succinctement décrit dans la section suivante (5.2). Les algorithmes de segmentation ayant montré qu'ils n'étaient pas assez robuste pour nos besoins, nous avons cherché une base différente, plus proche de l'image et plus robuste sur laquelle nous appuyer pour construire des structures plus complexes. Nous avons choisi de nous appuyer sur les algorithmes de détection de points caractéristique (*feature detection algorithm*). Cette famille d'algorithmes extrait d'une image un ensemble de points associés à un vecteur de caractéristiques, appelé descripteur. L'ensemble de points est choisi soit aléatoirement, soit sur une grille, soit à l'aide d'un détecteur de points d'intérêt, cherchant généralement les points de forts contrastes. Le vecteur de caractéristiques est calculé à partir d'une région de l'image autour du point. L'algorithme utilisé pour calculer ce descripteur à partir d'une région de l'image est ce qui fait la spécificité de chaque détecteur. Ces algorithmes sont conçus pour être robuste aux changements de luminosité et aux distorsions sur l'image (rotation, transformation affine, changement d'échelle, selon l'algorithme) : le vecteur de caractéristiques doit être peu modifié par de telles distorsions. La section 5.3 donne une liste des algorithmes que nous avons considérés.

Tous les algorithmes de cette famille peuvent être vus de manière homogène, comme prenant une image en entrée et donnant en sortie un ensemble de points associés à des vecteurs, ou plus généralement à des éléments d'un espace topologique associé à une fonction de distance. Contraindre la sortie à des vecteurs est toujours possible en rendant mathématiquement inexacts certains algorithmes comme nous le montrerons par la suite, mais permet une optimisation significative des calculs en aval avec en pratique une très faible dégradation.

Le calcul d'une distance entre deux vecteurs étant une opération coûteuse, nous avons choisi le modèle classique qui consiste à construire un dictionnaire de vecteurs caractéristiques, puis de ne plus utiliser que les mots du dictionnaire pour les calculs en aval. La section 5.4 est consacrée à la description des algorithmes de classifications utilisés pour construire ce dictionnaire. Nous nous sommes restreints aux techniques de classifications incrémentales, ce qui permet aux robots d'adapter leur dictionnaire de caractéristiques en cas de changement de l'environnement.

Les classes de descripteurs ne peuvent pas être utilisées dirrectement comme référents dans des jeux de langage : si les points caractéristiques sont détectés à des positions fixes dans l'image, les deux robots observant la même scène ne détecteront pas les mêmes, à moins de choisir un nombre de positions trop important pour les jeux de langage. Nous avons expérimenté avec divers algorithmes de détection de points d'intérêts, du plus simple [Harris and Stephens, 1988] au plus complexe [Itti et al., 1998] mais aucun ne s'est révélé assez robuste pour fournir les mêmes 5 à 10 points les plus salients sur deux images de la même scène d'un point de vue légèrement décalé.

En conséquence, nous utilisons ces points caractéristiques comme bases pour un modèle d'objet plus complexe. La section 5.5 est consacrée à ce modèle d'objet. Nous définissons un concept d'objet comme la co-occurrence de mots du dictionnaire de caractéristiques dans une même région de l'image et montrons comment des modèles d'objets peuvent être appris et modifiés, de manière incrémentale et non supervisée.

Les dernières sections de ce chapitre détaillent les résultats que nous avons obtenus avec ce modèle, sur des bases de données d'images d'abord, puis sur le *guessing game*.

5.2 La reconnaissance d'objets

Cette section a pour objectif de dresser un rapide état de l'art du domaine de la reconnaissance d'objets basée sur la vision et, plus particulièrement, les techniques utilisant des détecteurs de points d'intérêt et des vecteurs de caractéristiques.

En 1999, [Squire et al., 1999] proposent d'appliquer des techniques issues de l'indexation de documents textuels à la classification d'images. L'idée consiste à réduire une image à un ensemble de descripteurs, puis à discrétiser ces descripteurs au moyen d'une classification. L'image vue comme un ensemble de classes de descripteurs est alors analogue à un document vu comme l'ensemble des mots qui le composent. Les auteurs utilisent une représentation optimisée associant à chaque classe de descripteur les images dans lesquelles elle apparaît. Ils proposent un calcul de distance entre images utilisant une pondération basée sur la fréquence d'occurrence de chaque classe et construisent une base de données d'images capable de répondre à des requêtes constituées elles-même de fragments d'autres images.

L'émergence dans les années suivantes de descripteurs rapides à calculer et très robustes va permettre un développement important de cette voie de recherche.

[Sivic and Zisserman, 2003] développent un système similaire pour l'indexation de vidéos, en utilisant une version du descripteur SIFT (cf. 5.3.3) utilisant un détecteur d'ellipses d'intérêt augmentant la robustesse aux transformations affines. Le schéma général du système reste le même : un dictionnaire de classes de descripteurs d'environ 10 000 mots est créé à partir d'un sous-ensemble de la base de données, en utilisant l'algorithme des k-moyennes (k-means). A chaque image est associé un vecteur de taille 10 000 contenant le nombre d'occurrences de chacun des mots en présence de l'image. Chaque composante de ces vecteurs est pondérée par un terme appelé IDF (inverse-document frequency) égal au logarithme de l'inverse de la fréquence d'occurrence du mots dans le corpus : plus un mot est rare, plus son poids sera important. La similarité entre deux images est alors définie comme le produit scalaire de ces deux vecteurs pondérés, normalisés, qui est aussi le cosinus de l'angle entre ces deux vecteurs. Cette mesure de similarité est ensuite utilisée pour classer les images de la base en réponse à une requête.

De nombreux travaux récents proposent des versions différentes de chaque étape de ce modèle général (détection des points caractéristiques, création du dictionnaire de descripteurs, calcul de similarité entre images à partir de ce

dictionnaire).

Dance et al. [2004] présentent un système très similaire dans lequel un SVM (*support vector machine*) est utilisé pour apprendre des classes d'objets, en prenant comme entrée les vecteurs d'occurrences des mots du dictionnaire de descripteurs.

Construction du dictionnaire. [Jurie and Triggs, 2005] analysent l'étape de construction du dictionnaire de caractéristiques, réalisée par un *k-mean* dans la grande majorité des travaux et concluent que cet algorithme est inadapté à la grande hétérogénéité de la distribution des caractéristiques de grande dimension (comme SIFT de dimension 128), car il concentre la majorité des mots sur la zone la plus dense. Les auteurs montrent que les descripteurs les plus utiles sont ceux dont la fréquence d'occurrence est moyenne : les descripteurs trop fréquents ne sont pas assez discriminant et les descripteurs apparaissant trop rarement, même s'ils sont discriminant, sont trop rares pour peser significativement sur le système global. Les auteurs proposent un algorithme itératif qui recherche à chaque étape la boule de rayon fixé contenant le plus grand nombre de descripteurs, crée un mot à cette position et supprime les descripteurs contenus dans cette boule avant de recommencer. Les auteurs montrent aussi que la détection dense des descripteurs SIFT donne de meilleurs résultats que la détection sur des points d'intérêt.

[Nistér and Stewénius, 2006] proposent une version hiérarchique du *k-mean*, plus rapide et qui peut être réalisée de manière incrémentale. Le classifieur est un arbre, chaque nœud ayant k fils. Lorsque le système décide d'étendre une feuille de l'arbre, un *k-mean* est réalisé sur tous les descripteurs associés à cette feuille et un sous-nœud est associé à chacun des k centres de classe obtenus. Chaque descripteur est associé au nœud dont le centre est le plus proche (segmentation de Voronoï). Cette méthode permet aussi une recherche rapide du ou des mot associés à un descripteur.

[Moosmann et al., 2006b] utilisent comme classifieur un ensemble d'arbres générés semi-aléatoirement. Chaque niveau de chacun des arbres coupe l'espace vectoriel du descripteur en deux. Le plan de découpage est choisi aléatoirement, mais seuls les découpages vérifiant une contrainte entropique sont conservés. A chaque descripteur est donc associée une feuille de chacun des arbres. Les auteurs utilisent 5 arbres à 1000 feuilles chacun. La sortie de ces arbres est utilisée pour apprendre un SVN.

Recherche dans le dictionnaire. Rechercher le mot associé à un descripteur donné est une opération réalisée très fréquemment par les systèmes de reconnaissance d'objets. La méthode naïve consiste à tester chacun des mots du dictionnaire. Cette méthode se révèle trop coûteuse pour fonctionner en temps réel dès que le nombre de mots dépasse quelque milliers, surtout pour les descripteurs de dimension élevée, pour lesquels le calcul de distance est lui-même coûteux.

L'alternative utilisée dans la majorité des travaux de ce domaine utilise une représentation hiérarchique du dictionnaire sous la forme d'un *kd-tree*. Cette structure permet de faire une recherche du plus proche voisin exacte ou approximative très rapide.

Modèles de classification probabilistes. [Sivic et al., 2005] proposent d'appliquer un modèle probabiliste issue de l'indexation de textes appelé PLSA (pour *Probabilistic Latent Semantic Analysis*) et capable de découvrir des classes d'objets de manière non supervisée. Cette méthode modélise chaque image comme une combinaison pondérée d'objets (topics) et utilise un algorithme de maximisation d'espérance (EM) itératif pour les découvrir. Les auteurs montrent que le résultat de cette classification peut être utilisée ensuite pour segmenter les objets présents dans les images.

[Huei-Ju Chen and Triesch, 2005] utilisent une autre méthode probabiliste en modélisant par un réseau Bayesien représentant des hypothèses sur la reconnaissance et la segmentation des objets. Une méthode d'inférence probabiliste basée sur le Gibbs sampling (approche type Monte Carlo) est utilisée pour construire le réseau.

[Fei-Fei et al., 2006] utilise un modèle bayésien capable d'apprendre un modèle d'objet à partir seulement d'une poignée d'exemples et fonctionnant avec une centaine de catégories d'objets.

Méthodes de vote. [Murphy-Chutorian et al., 2005] proposent un système dans lequel chaque descripteur vote de manière pondérée pour les modèles d'objets. Les auteurs développent un algorithme pour évaluer la pondération et les seuils de détection optimaux.

[Filliat, 2007] développe un système à base de vote à deux niveaux pour identifier la pièce dans laquelle se trouve un robot. Tout d'abord, chaque descripteur vote pour les modèles d'objets pour lesquels il a été vu. Puis les votes des différents algorithmes sont agrégés. Ce système présente la par-

ticularité d'être capable de ne pas donner de réponse si l'image présentée n'est pas significative, ce qui lui permet d'obtenir au taux de faux positifs quasi-nul.

Utilisation de l'information spatiale. Certains systèmes étendent le modèle sac-de-mots issu de l'indexation de textes, qui ne considère une image que comme l'ensemble de ses descripteurs sans information spatiale.

[Jodogne et al., 2005] utilisent en plus des classes de descripteurs élémentaires des classes composites, composées d'une paire de classes de descripteurs et d'une distance. Les composantes des classes composites peuvent elles-mêmes être des classes composites. Ces classes composites sont créées en sélectionnant des paires de mots dont la co-occurrence est fréquente, puis en utilisant l'histogramme de leurs distances sur toutes les images pour choisir l'intervalle de taille fixée le plus probable. L'intérêt de cette approche est que le système de classification d'objets utilise indifféremment les classes élémentaires et les classes composites.

[Quack et al., 2006] proposent un modèle dans lequel un descripteur composite est formé pour un descripteur simple donné en prenant les 5 descripteurs les plus proches dans l'image et en les associant avec le quadrant de l'image dans lequel ils ont été détectés.

[Lazebnik et al., 2006] développent une représentation d'une image en la recouvrant d'une pyramide de fenêtres de tailles décroissantes et en associant à chaque niveau l'histogramme des descripteurs contenus dans la fenêtre. Les auteurs utilisent simultanément SIFT et un descripteur plus simple mais moins discriminant, associé à un détecteur de points caractéristiques simple pour obtenir un nombre plus important de points.

5.3 Algorithmes de détection de points caractéristiques

Cette section décrit les quatre algorithmes de détection de caractéristiques que nous avons testés, en rappelant brièvement la nature de la caractéristique, l'algorithme utilisé pour choisir les points sur lesquels elle est calculée, ses invariants et la complexité de la détection.

5.3.1 Histogramme et distance EMD

L'histogramme est une caractéristique simple et facilement calculable. Il peut être utilisé sur une ou plusieurs composantes de n'importe quel espace de couleurs, à divers niveaux de discrétisation. On le trouve typiquement utilisé dans la littérature sur la composante Hue de l'espace HSV, ou sur le niveau de gris, avec une discrétisation assez grossière (16 valeurs). Il est calculé sur des fenêtres, typiquement choisis de manière dense et à plusieurs résolutions dans l'image. Les distances vectorielles sont peu adaptées pour le calcul de distance entre deux histogrammes, un point passant d'un secteur de l'histogramme au secteur adjacent donnant la même distance que ce même point passant sur le secteur opposé. Les deux distances les plus utilisées sont les distances EMD [Rubner et al., 1998] et la distance de diffusion [Ling and Okada, 2006], qui toutes les deux calculent l'*effort* nécessaire pour passer entre les deux histogrammes, en prenant en compte leur structure. L'histogramme est relativement robuste aux rotations, ne conservant aucune information spatiale, et aux changements de luminosité (si appliqué sur les composantes Hue et Saturation de HSV).

5.3.2 Corrélogramme

Les corrélogrammes introduits en traitement de l'image par [Huang et al., 1997] sont un raffinement des histogrammes conservant l'information spatiale, ce qui leur confère un plus grand pouvoir discriminant. Leurs caractéristiques sont très proches des histogrammes, hormis un calcul légèrement plus coûteux.

5.3.3 SIFT

SIFT (*Scale Invariant Feature Transform*) est un descripteur introduit par David Lowe dans [Lowe, 2004], invariant en rotation et changement d'échelle. Le descripteur est de taille 128 et composé d'histogrammes d'orientations sur quatre fenêtres autour d'un point. Il est associé à l'orientation et l'échelle à laquelle il a été détecté. Les points sur lesquels ce descripteur est calculé sont choisis en filtrant l'image par un filtre Gaussien à différentes échelles puis en extrayant les extrêmes en espace et en échelle. Ce descripteur est l'un des plus utilisés dans la littérature. Plusieurs implantations en sont disponibles, les plus utilisées étant celle fournie par D.Lowe

sous forme binaire uniquement et Siftpp, développé à l'UCLA (http ://vision.ucla.edu/~vedaldi/code/siftpp/siftpp.html). Nos tests montrent que l'implantation de Lowe est plus performante : une fois les deux implantations configurées pour obtenir le même nombre de points, le taux de points mis en correspondances correctement entre deux images proches de la même scène est sensiblement supérieur pour cette dernière. Pour les images de l'aibo, de résolution 208x160, l'algorithme met environ 300 millisecondes sur un pentium4 à 2GHz pour extraire environ 150 vecteurs SIFT.

5.3.4 K-Adjacent Segment

K-AS est un descripteur introduit dans [Ferrari et al., 2006] : l'image est tout d'abord traitée par un détecteur de contours. Ces contours sont ensuite réduits à un ensemble de segments en isolant les parties "presque droites". Puis un graphe est construit en reliant les segments ayant leurs extrémités proches. Un descripteur est alors calculé à partir de chaque sous-ensemble connexe de k points de ce graphe. Ce descripteur de dimension 4k-2 est invariant aux changements d'échelle et stocke les orientations relatives des segments les uns par rapport aux autres, ainsi que la distance entre leurs milieux, relative à leurs tailles. Ce descripteur est très compact par rapport aux trois exposés ci-dessus, mais il utilise une information de plus haut niveau, à savoir des contours et non l'image brute. Les auteurs montrent que les meilleurs résultats sont obtenus pour k=2, et obtiennent de très bonnes performances de reconnaissance, sur des bases de données d'objets dont la forme est très discriminante. L'algorithme k-as est relativement lent, de l'ordre de la minute, principalement à cause du détecteur de contours.

5.4 Graphes de discrimination

Afin d'accélérer les calculs, les systèmes de vision artificielle basés sur des vecteurs de caractéristiques utilisent généralement un dictionnaire de caractéristiques de quelques milliers de mots : à chaque vecteur est associé le ou les mots les plus proches du dictionnaire et seuls ces mots sont utilisés par la suite. Cette discrétisation permet aussi de se retrouver dans un cadre plus propice à des calculs probabilistes.

La méthode classiquement employée pour construire ce dictionnaire est d'utiliser un algorithme *off-line* de classification sur un échantillon des car-

actéristiques, comme par exemple un *k-mean*. Cette méthode requiert de disposer *a priori* d'un échantillon représentatif des vecteurs de caractéristiques. Comme nous souhaitons que les *Talking Robots* soient capables de s'adapter aux changements de leur environnement, cette méthode *a priori* ne nous convient pas : si le dictionnaire devient inadapté à l'environnement, il ne communiquera pas les discriminations pertinentes entre caractéristiques au niveau de calcul supérieur, ce qui dégradera ses performances.

Les propriétés que nous recherchons pour notre dictionnaire sont :

- Construction incrémentale adaptative : Le dictionnaire doit pouvoir créer de nouveaux mots au besoin en fonction des caractéristiques qui lui sont présentées.
- Vocabulaire hiérarchique : Nous ne savons pas *a priori* quel niveau de discrimination sera adéquat. Un vocabulaire à un seul niveau nécessiterait un paramétrage fin pour chaque algorithme de détection de caractéristiques. Un vocabulaire hiérarchique permet de communiquer au niveau supérieur des mots à différents niveaux de spécificité, en le laissant déterminer lesquels sont pertinents (par exemple en fonction de leur fréquence d'occurrence). De plus, une structure hiérarchique permet d'optimiser la recherche dans le dictionnaire en évitant de devoir parcourir tous les nœuds.
- Permanence des mots : Une fois un mot créé, sa définition ne doit plus évoluer.
- Stabilité : Le dictionnaire ne doit pas se dégrader ou grossir infiniment avec le temps.

On observe sans surprise que les arbres de discrimination (cf. 2.4) répondent à des besoins similaires dans le cadre perceptif des *Talking Heads*.

Nous avons expérimenté avec deux structures de dictionnaire différentes, dans les deux grandes catégories de structures hiérarchiques possibles : celles à rayon fixé et celles à nombre de fils ou nombre de caractéristiques par nœud fixé [Omohundro, 1989].

5.4.1 Premier dictionnaire : taille de rayon fixe

Nous avons basé notre première structure de discrimination hiérarchique sur une structure arborescente avec un rayon de classifieur fixe par niveau de profondeur. Notre structure est définie comme suit :

- A chaque nœud i sont associés une profondeur P_i égale à la longueur du chemin le séparant de la racine, un point de l'espace de caractéristiques

C_i et un rayon $R_i = f(P_i)$, f étant une fonction décroissante (nous avons utilisé $x \rightarrow K^x$ avec $K \in]0,1[$.

- Il y a un arc de i vers j si et seulement si $P_j = P_i + 1$ et si le cluster de j est inclus dans le cluster de i, ce qui se traduit pratiquement en utilisant l'inégalité triangulaire par $i \rightarrow j \; ssi \; D(C_i, C_j) < R_i - R_j$.
- Une profondeur maximum, soit un rayon de classifieur minimum, est fixée par avance.

On constate comme illustré figure 5.1 qu'avec cette définition, un nœud peut avoir plusieurs parents.

Par construction, si une caractéristique n'est pas contenue dans un des nœuds, alors elle ne sera contenue dans aucun de ses enfants, ce qui permet une recherche rapide de tous les nœuds contenant une caractéristique donnée.

Lorsque une nouvelle caractéristique est présentée au dictionnaire, un nouveau nœud est créé à chaque niveau de profondeur pour lequel la caractéristique n'est contenue dans aucun nœud existant. Ce nouveaux nœud est créé avec comme centre la nouvelle caractéristique, comme rayon le rayon de la profondeur en question et comme parent l'ensemble des nœuds de la profondeur supérieure le contenant.

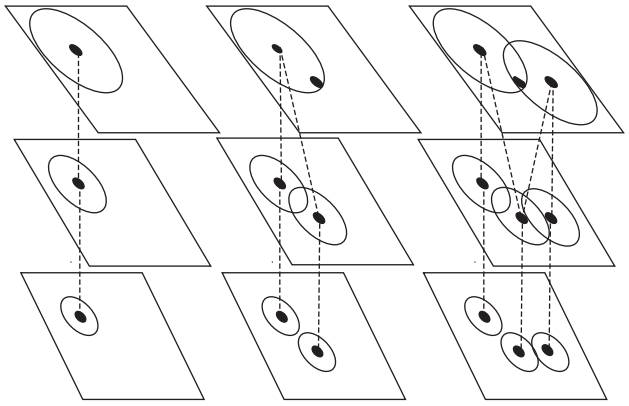


FIG. 5.1 – Exemple de construction d'un graphe de discrimination.

L'intérêt de ce système est qu'il se contente d'un espace topologique sur les caractéristiques : disposer d'un espace vectoriel n'est pas nécessaire.

Mais les performances obtenues par ce système ne sont pas satisfaisantes : les espaces vectoriels de grande dimensions, comme celui des caractéristiques SIFT (dimension 127) se prêtent mal à ce type de structures. En effet le

rapport du volume d'une hypersphère sur le volume de l'hypercube dans lequel elle est inscrite décroît et tend vers 0 quand la dimension augmente. Ce qui veut dire que plus la dimension est importante, plus il faut d'hypersphères pour remplir l'espace. Notre graphe dégénère en une structure ou chaque niveau de profondeur est entièrement connecté au suivant, ce qui le rend inutilisable.

5.4.2 Second dictionnaire : nombre de fils par nœud fixe.

Comme nous venons de le montrer, partitionner l'espace en sous-espaces de même taille n'est pas adapté pour les grandes dimensions. L'alternative consiste à partitionner l'espace en un nombre de sous-espaces fixé, sans se soucier de leurs tailles relatives. La méthode que nous avons utilisée est une simplification de l'algorithme de *k-mean* hiérarchique [Nistér and Stewénius, 2006]. Chaque nœud intermédiaire de l'arbre divise le sous-espace qu'il représente en deux. La création de deux nouveaux nœuds à partir d'une feuille intervient lorsque le nombre de caractéristiques rencontrées contenues dans le sous-espace qu'elle représente dépasse un seul fixé. Pour permettre le calcul de la "meilleure" séparation, chaque feuille garde un échantillon de 100 caractéristiques choisies aléatoirement parmi celles rencontrées.

Nous n'avons pas trouvé de solution satisfaisante permettant de calculer une partition "efficace" de l'espace en nous limitant à une fonction de distance sur les caractéristiques. En effet, tous les calculs dans ce cadre nécessitent de disposer de la matrice des distances entre toutes les paires de points de l'échantillon, ce qui ne peut pas être obtenu en temps-réel. Nous avons expérimenté avec l'alternative sans calculs : à chaque sous-nœud à créer est attribué aléatoirement une caractéristique de l'échantillon et la partition est réalisée par la segmentation de Voronoï basée sur ces deux caractéristiques. Mais les performances obtenues avec un arbre généré de cette manière varient énormément d'une itération sur la suivante. Dans [Moosmann et al., 2006b], les auteurs utilisent plusieurs arbres générés aléatoirement en parallèle, mais ils font passer la sortie de ces arbres à travers un SVM, ce qui rend la construction des arbres impossible à réaliser en ligne.

Nous avons donc imposé à nos caractéristiques la contrainte d'être dans un espace vectoriel. Dans ce cadre, nous avons retenu l'algorithme de partition utilisé dans [Oudeyer et al., 2007] : la partition est réalisée par l'hyperplan

orthogonal à un des axes minimisant la somme des variances pondérées des deux classes. Ce critère a l'avantage de pouvoir être calculé rapidement, en $O\left(d.nlog(n)\right)$ où d est la dimension de l'espace et n le nombre de points dans l'échantillon. En effet, pour chaque axe, il suffit de trier les n points selon cet axe. Il y a alors $n-1$ partitions possibles à tester et le calcul de la variance peut être fait de manière incrémentale : chaque nouvelle partition testée ne diffère de la précédente que par un point qui change de côté.

Les performances obtenues avec cet algorithmes sont meilleures et plus stables qu'avec une partition aléatoire (cf section 5.10).

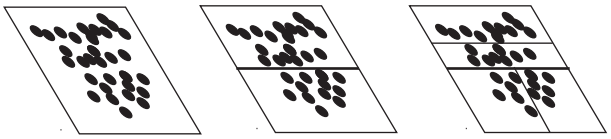


FIG. 5.2 – Exemple de construction d'un arbre de discrimination.

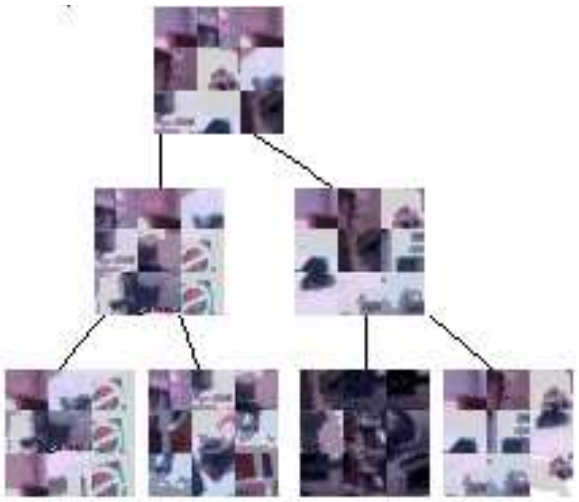


FIG. 5.3 – Portion d'arbre de discrimination pour SIFT.

5.5 Les objets

5.5.1 Définition

La couche d'arbres de discrimination a réduit l'information contenue dans l'image à un ensemble de coordonnées, chacune associée à une liste de mots du dictionnaire. L'objectif de la couche "objet" est de construire de manière incrémentale, non supervisée et en temps réel des modèles d'objets à partir de la sortie des arbres de discrimination. Ces modèles seront les abstractions utilisées par les agents pour communiquer.

Le sens donné ici au terme "objet" n'est pas aussi abstrait que son usage courant pourrait le faire penser. Un tel concept est hors de la portée d'un robot aux capacités d'action limitées comme l'aibo et ne pourrait être extrait par un système basé uniquement sur la vision. Nous définissons ici un objet comme une co-occurrence stable de plusieurs primitives visuelles dans une zone de la perception. Nous pensons que ce concept très proche de l'image pourra permettre aux aibos d'identifier et de communiquer sur des zones de l'espace les entourant et pourra servir de bases à des communications plus abstraites dans de futures expériences.

Notre modèle est du type sac de mots (*bag of words*) : la position relative des différentes caractéristiques est ignorée : seule l'information de présence ou d'absence d'une caractéristique est utilisée. Cette classe de modèles issue de la recherche en indexation de documents textuels a seulement récemment été appliquée à l'analyse d'images [Squire et al., 1999]. De nombreux résultats (cf. section 5.2) montrent que cette information réduite dispose d'une capacité de discrimination suffisante dans de nombreux cas de figure. Si cela s'avère nécessaire, l'information de position pourra être rajoutée à notre modèle par la suite, sous la forme d'une nouvelle caractéristique composée portant sur des relations spatiales entre caractéristiques primitives.

Nous définissons un modèle d'objet comme un vecteur de taille égale au nombre de mots dans le dictionnaire. Ce vecteur contient le nombre de fois où chaque mot du dictionnaire a été vu en présence d'un objet identifié comme correspondant au modèle.

Avec cette définition, on constate qu'un nouveau modèle d'objet peut être créé à partir d'une image seule, puis mis à jour à chaque nouvelle image lui correspondant en sommant simplement les vecteurs.

5.5.2 Comparaison

Pour décider si une image appartient ou non à un modèle, il nous faut tout d'abord être capable de comparer une image à un modèle existant, soit deux modèles entre eux. Nous avons expérimenté avec diverses fonctions de distances, détaillées dans les sous-sections suivantes. Toutes ces distances utilisent une version normalisée du vecteur d'occurrences, afin de pouvoir comparer des modèles mis à jour un nombre différent de fois. Nous avons choisi de normaliser sur le vecteur et non de le diviser par le nombre de mises à jour pour ne pas être sensible aux variations de nombre de caractéristiques vues dans chaque image.

5.5.2.1 Normes L1 et L2, TF-IDF

Il est bien sùr possible d'utiliser les normes L1 et L2 directement sur les vecteurs d'occurrences, mais les performances sont alors très mauvaises. En effet les éléments du vecteur correspondant à des mots plus génériques du dictionnaire ont une valeur plus importante que ceux correspondants à des mots plus spécifiques, ce qui fait que les mots spécifiques ont peu de poids dans le calcul : la représentation vectorielle ne rend pas compte des relations d'inclusions entre les différents mots.

La solution utilisée couramment consiste à pondérer chaque élément par une fonction appelée *term-frequency, inverse document frequency* (TF-IDF) [Sivic and Zisserman, 2003]. La formule utilisée est $w_{di} = p_{di} log\left(\frac{1}{p_i}\right)$. w_{di} est la nouvelle composante i du vecteur d, p_{di} son ancienne valeur (la probabilité qu'un mot soit i pour l'objet d) et P_i la valeur de cet élément dans le vecteur obtenu en sommant tous les objets, soit la probabilité qu'un mot soit i dans le corpus de tous les objets. Cette pondération donne plus d'importance aux mots apparaissant moins fréquemment. Une fois appliquée, la comparaison s'effectue avec la norme L1 ou L2. Le vecteur $(p_i)_i$ pouvant être calculé de manière incrémentale à chaque modification d'un des modèles d'objets, ce calcul n'est pas plus coûteux que la norme sous-jacente. Nos tests hors-lignes (cf 5.10.2) ont montré que la distance TF-IDF était celle donnant les meilleures performances. C'est cette distance qui sera utilisée par la suite.

5.5.2.2 Calcul probabiliste

Chaque élément i du vecteur d'occurrences normalisé $(p_{di})_i$ du modèle d peut être vu comme la probabilité qu'un mot perçu en présence du modèle d

soit i. Dans ce cadre, étant donné un modèle m calculé à partir d'une nouvelle image, on peut calculer la probabilité que cette image soit une occurrence du modèle d par la règle de Bayes :

$$p\left(d|m\right) = p\left(m|d\right)\frac{p\left(d\right)}{p\left(m\right)}$$

En faisant l'hypothèse qu'on sait inexacte que les mots du dictionnaire sont indépendants, on obtient :

$$p\left(d|m\right) = \left(\prod_i m_i d_i\right) p\left(d\right)$$

Le terme $p\left(m\right)$ peut être ignoré car seuls nous intéressent les probabilités relatives pour les différents modèles.

Algorithmiquement le logarithme de cette expression est calculé, pour éviter de dépasser la capacité de représentation des types flottant.

Les performances obtenues dans notre cadre avec cette distance sont mauvaises. L'utilisation d'un dictionnaire hiérarchique rend en effet l'hypothèse d'indépendance des mots complètement fausse.

5.5.2.3 Vote

Comparer un modèle construit à partir de plusieurs images avec un modèle construit à partir d'une seule image pose un problème : le premier modèle reflète la probabilité de percevoir chaque mot, alors que le second reflète un tirage, correspondant ou non à cette probabilité. Par exemple, si un mot a été vu une fois sur deux lors de l' apprentissage d'un modèle et que son poids normalisé est k, alors une image de ce modèle aura un poids pour ce mot de 0 ou d'une valeur proche de $2k$ selon que la caractéristique correspondant à ce mot est visible ou non (toutes choses étant égales par ailleurs).

Les distances par vote fréquemment utilisées dans les systèmes de perceptions à base de caractéristiques [Filliat, 2007] essayent de régler ce problème. Le vecteur d'occurrence est remplacé par un vecteur binaire en le seuillant à 1 et la distance L1 est utilisée sur ces vecteurs.

Malheureusement, cette distance a pour effet de perdre l'information de proportion entre les différents mots pour un modèle donné et l'effet dans notre cadre est globalement négatif : les performances sont plus mauvaises que TF_IDF.

5.5.2.4 Pondération entropique

Il est possible à l'aide d'un calcul entropique d'attribuer à chaque mot un poids proportionnel à son pouvoir discriminant :

$$h_i = -\sum_d p_{di} log\left(p_{di}\right)$$

Cette quantité positive est minimale pour les mots les plus discriminants, c'est-à-dire ceux pour lesquels la distribution de probabilité sur tous les modèles d'objets est la plus hétérogène. Nous avons expérimenté en divisant le vecteur d'occurrence par h, en remplaçant ses composantes nulles par la plus petite valeur possible $\frac{1}{n} log\left(\frac{1}{n}\right) + \frac{n-1}{n} log\left(\frac{n-1}{n}\right)$ où n est le nombre de mots du dictionnaire.

Nos expérimentations ne montrent pas d'amélioration significative des performances en utilisant la pondération entropique. La variation des scores d'entropie varie en effet peu d'un mot à l'autre dès que le nombre de modèles devient important.

5.5.3 Reconnaissance, construction non supervisée

Dans le cadre des *Talking Robots*, notre objectif est que chaque agent puisse créer et maintenir une liste de modèles d'objets à partir de ses perceptions, de manière incrémentale et non supervisée.

Maintenant que nous disposons d'un moyen de comparer deux modèles, il nous reste à trouver un critère pour décider si l'agent doit intégrer une nouvelle image au modèle le plus proche existant, ou créer un nouveau modèle.

Nous ne pouvons pas utiliser un seuil absolu fixé arbitrairement, car les ordres de grandeurs des distances entre modèles dépendent de nombreux paramètres (nombre de mots du dictionnaire, détecteurs de caractéristiques utilisés, etc.).

Nous avons opté pour une solution proche de celle retenue dans [Lowe, 2004] pour le critère de décision d'appariement de deux caractéristiques. Dans ce modèle, Lowe cherche à apparier les caractéristiques de deux images différente. Pour chaque caractéristique de la première image c_i^1, il cherche les deux caractéristiques les plus proches dans la deuxième image, c_j^2 et c_k^2. Il décide d'apparier c_i^1 à c_j^2 si $\frac{d\left(c_i^1, c_j^2\right)}{d\left(c_i^1, c_k^2\right)} \geq K$, K étant une constante qu'il fixe à $1,2$.

Nous étendons ce critère en comparant la distance de l'image au meilleur modèle à la moyenne des distances de l'image aux N modèles suivants. L'image est intégrée au meilleur modèle si ce rapport est supérieur à K. Nous utilisons typiquement N entre 5 et 15. Cette extension permet d'éviter au système de dégénérer si deux modèles se retrouvent proches l'un de l'autre. Une meilleure solution à ce problème serait de fusionner deux modèles lorsque ceux-ci deviennent trop proches, mais cette fonctionnalité n'a pas été intégrée à notre système.

La constante K permet de régler la spécificité des modèles : plus elle est basse, plus des images éloignées seront intégrées au modèle.

L'utilisation du système se déroule comme suit. Chaque nouvelle image est comparée à tous les modèles existants. Si le critère d'intégration est vérifié, l'image est intégrée au meilleur modèle. Sinon, un nouveau modèle est créé, composé de cette seule nouvelle image. Naturellement, le système va en permanence créer de nouveaux modèles. Nous avons choisi de fixer un nombre de modèles maximal. Lorsque ce nombre est atteint et qu'un nouveau modèle doit être ajouté, le modèle le moins fréquemment rencontré, d'un âge supérieur à un seuil fixé (cf 5.10.1) est supprimé. Afin de limiter la fréquence de création de nouveaux modèles, l'ajout d'un nouveau modèle ne se fait pas systématiquement, mais avec une probabilité $p < 1$ fixée).

L'initialisation des modèles se fait en présentant au système des images différentes (typiquement une dizaine). Un modèle est alors créé à partir de chaque image.

5.5.4 Agrégation de multiples détecteurs de caractéristiques

Notre système d'objet peut utiliser des mots provenant de plusieurs dictionnaires et donc issus de plusieurs détecteurs de caractéristiques différents. Le vecteur d'occurrences est alors la concaténation des vecteurs d'occurrences pour chaque dictionnaire. Cette solution simple présente une importante limitation : le nombre de mots de chaque dictionnaire et le nombre de caractéristiques détectées dans l'image doivent être du même ordre de grandeur pour chaque détecteur, sous peine de voir certains détecteurs en "écraser" d'autres dans le calcul de distance entre modèles. Nous avons brièvement expérimenté avec des systèmes d'agrégation plus complexes, en effectuant un calcul de distance séparé pour chaque détecteur, puis en agrégeant par

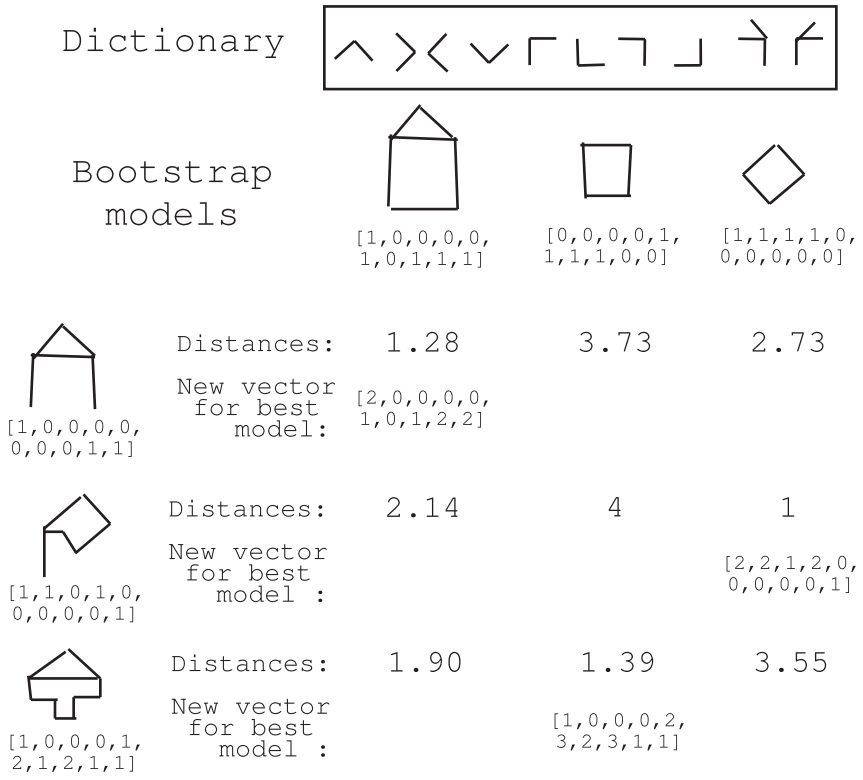
moyenne pondérée, ou par vote, sans aboutir à une solution satisfaisante.

Dans la suite de nos expériences, nous utilisons uniquement le détecteur SIFT, qui se révèle être à la fois le plus performant et le plus rapide.

5.5.5 Fenêtrage

Afin de détecter des co-occurrences de caractéristiques stables de taille inférieure à toute l'image, l'algorithme de reconnaissance et de création de modèles d'objets peut être appliqué sur un ensemble de fenêtres superposées les unes aux autres. Pour les expériences avec l'aibo, nous avons utilisé 16 fenêtres de taille 104x80 se superposant sur la moitié de leur taille (la résolution de la caméra de l'aibo étant 208x160). Dans cette configuration, un modèle d'objet est détecté ou créé indépendamment sur chaque fenêtre.

Cette solution simple est loin d'être optimale. Forcer la détection ou la création systématique d'un modèle sur chaque fenêtre alors que celles-ci se recouvrent partiellement conduit à créer de nombreux modèles proches les uns des autres, ou alors composés uniquement de "bruit". Une solution meilleure, mais plus complexe serait de mettre en compétition les fenêtres les unes envers les autres par un mécanisme d'inhibition entre fenêtres adjacentes. Le système essayerait alors d'*expliquer* au mieux la présence des caractéristiques perçues par les modèles d'objet.



La norme utilisée est L1 sur les vecteurs normalisés.

FIG. 5.4 – Exemple simple d'évolution de modèles d'objets.

5.6 Mots et lexique

Les signifiants sur lesquels les agents vont échanger des mots sont les modèles d'objets décrits dans la section précédente.

Chaque agent dispose en plus de ses modèles d'objets d'un lexique composé de triplets (modèle, mot, poids) initialement vide.

Les mots sont des combinaisons aléatoires de 4 syllabes choisis dans une liste, comme dans la première version des *Talking Robots*.

5.7 Initialisation : jeu de discrimination

Afin d'accélérer les expériences, chaque agent initialise ses arbres de discrimination et ses modèles d'objets seul avant le *guessing game*. Cette phase peut s'apparenter au jeu de discrimination des *Talking Heads* et de notre première version des *Talking Robots*.

Chaque robot se déplace aléatoirement dans son environnement et fait l'acquisition d'une nouvelle image en tournant sa tête dans une direction aléatoire à intervalle constant. Les arbres de discrimination sont grossis jusqu'à atteindre une taille fixée à l'avance (cf. 5.10.1) . Puis l'agent continue à acquérir des images, qui sont utilisées pour développer les modèles d'objets. Une fois un nombre d'itération fixe passé, choisi pour que le nombre maximum de modèles autorisés soit atteint, les modèles d'objets sont nettoyés en supprimant les 10% vues les moins souvent.

5.8 Guessing Game

Le *guessing game* se déroule de manière similaire à ses versions des *Talking Heads* et des premiers *Talking Robots*. Avant chaque itération, les deux agents se localisent et se positionnent côte-à-côte, en regardant dans la même direction. Chaque agent extrait de l'image le meilleur modèle d'objet sur chacune des 16 fenêtres. Le *speaker* choisit aléatoirement un modèle parmi ceux qui ne sont visibles que dans au plus quatre fenêtres adjacentes formant un carré, afin que la réussite du jeu soit statistiquement significative. Il cherche ou crée dans son lexique le meilleur mot associé à ce modèle et le communique au *hearer*.

Le *hearer* parcourt dans l'ordre tous les modèles de son lexique associés à ce mot, jusqu'à en trouver un ayant été détecté dans au moins une fenêtre. Il choisit alors la fenêtre avec le meilleur score de détection (la plus petite distance) pour ce modèle, puis pointe en son centre.

Le *speaker* vérifie la fenêtre pointée par le *hearer*. Si elle contient le modèle qu'il avait choisi, il signale un succès et l'itération se termine. Sinon, il pointe la meilleure fenêtre contenant ce modèle et signale un échec. En cas d'erreur, le *hearer* rajoute une entrée dans son lexique, associant le mot prononcé par le *speaker* au modèle d'objet perçu dans la fenêtre qu'il a pointé.

A l'issue de chaque itération, les pondérations des entrées du lexique sont modifiées comme pour les *Talking Heads* : En cas de succès le poids

de l'entrée du lexique utilisée est augmentée, le *speaker* diminue le poids de tous les homonymes et le *hearer* le poids de tous les synonymes. L'itération confirme en effet au *speaker* que le mot employé désigne le même modèle pour les deux agents et donc que ce mot ne doit pas être employé avec d'autres significations. L'itération confirme au *hearer* que le mot employé est le mot favori du *speaker* pour désigner le modèle, et donc que les autres mots ne doivent pas être employés pour ce modèle. En cas d'échec le poids des associations employées par le *hearer* et par le *speaker* sont diminués.

5.9 Implantation

Cette seconde version des *Talking Robots* est implantée moitié C++, moitié Urbi, ce dernier ayant atteint une maturité suffisante pour jouer ce rôle. Les détecteurs de caractéristiques sont implantés en réutilisant du code existant autant que possible, tous sous une même interface. Les calculs de distances entre vecteurs sont optimisés en utilisant sse2. La couche de détection d'objets est implantée en C++, Urbi n'étant pas adapté à la manipulation de grosses structures de données. Les primitives de comportement des *Talking Robots* (rechercher le meilleur modèle sur chaque fenêtre, choisir un modèle, recherche dans le lexique, etc) sont implantés en C++ et rendus accessibles en Urbi en utilisant l'API UObject. Le comportement global des agents est implanté en Urbi, qui permet des modifications rapides, ce qui facilite le prototypage d'expériences. Le code actif dans la dernière version des *Talking Robots* pèse environ 20.000 lignes de C++ et 3.000 lignes d'Urbi.

Toutes les étapes des différents calculs sont extrêmement verbeuses et la sortie de chaque expérience est conservée, ce qui permet d'extraire *a posteriori* toutes les informations nécessaires à l'analyse des résultats.

5.10 Résultats

Cette section présente les résultats que nous avons obtenus avec cette seconde version des *Talking Robots*, d'abord sur des bases de données d'images, puis dans un cadre d'apprentissage supervisé d'objets et enfin lors du *guessing game*.

5.10.1 Paramétrage

Malgré nos efforts pour en limiter le nombre, notre système présente de nombreux paramètres. Pour les régler, nous avons utilisé 3 classes de la base de données caltech101 [Fei-Fei et al., 2004](strawberry, panda, elephant), sur une tâche classique de classification supervisée. Les images sont divisées en un jeu d'apprentissage et un jeu de test de taille égale. Les arbres de discrimination sont initialisés à partir du jeux de test, en faisant passer les images dans un ordre aléatoire. Puis un unique modèle d'objet par classe est appris à partir de ce même jeu. Le taux de reconnaissance et la matrice de confusion sont calculés à partir du jeu de test.

Algorithme de détection de caractéristique et distance. Nous avons abandonné l'algorithme K-AS, utilisant un détecteur de contour, trop lent pour nos besoins (plus de 30s par image).

SIFT s'est révélé être le plus rapide et le meilleur des algorithmes restant (histogrammes, corrélogrammes). Le coupler avec un autre algorithme n'apporte pas d'amélioration notable sur la reconnaissance.

Parmi les distances listées en 5.5.2, la norme L2 de la pondération TF-IDF est d'après nos tests la meilleure.

Taille des arbres de discrimination. Nous avons effectué le test précédent en faisant varier le nombre de nœuds de l'arbre de discrimination (en refaisant passer plusieurs fois les mêmes images si nécessaire). Le taux de reconnaissance augmente avec le nombre de nœuds jusqu'à environ 1000 nœuds puis stagne au delà de cette valeur.

Modèles. Nous avons choisi le nombre maximum de modèles pour l'apprentissage non supervisé en fonction de la richesse de l'environnement. Nos *guessing game* se déroulant dans une seule pièce, nous avons fixé la limite à 150 objets, en nous basant sur le nombre de fenêtres visibles par un aibo immobile tournant sur lui même (360 degrés horizontaux, 60 verticaux pour 16 fenêtres par image et un champ de vision de 50x40 degrés donnent environ 170 fenêtres).

Les deux paramètres régulant la création et la suppression des objets (probabilité de création, période de sursis) sont étroitement liés à la fréquence attendue de création de nouveaux modèles. Nous avons observé que même une fois le nombre maximum de modèles atteint, le système crée en moyenne

un nouveau modèle sur deux des seize fenêtres par image. Ainsi, à titre d'exemple, si la probabilité de création est de 1 et que le temps de sursis d'un nouveau modèle est de 20 itérations, il y a en permanence 40 modèles récents "indestructibles" sur les 150, forçant les destructions de modèles à choisir parmi les 110 restants, plus anciens.

Pour avoir à la fois une quantité de modèles "utile" suffisante et un temps de sursis suffisant pour laisser au système l'opportunité de voir à nouveau les nouveaux objets avant de les supprimer, nous avons fixé la probabilité de création d'un nouveau modèle à 0.3 et le temps de sursis à 10 itérations.

5.10.2 Catégorisation sur la base d'images GRAZ-02

Dans un premier temps, nous avons voulu tester les capacités de catégorisation de notre système sur une base de données d'images.

La base d'image GRAZ-02 [Opelt et al., 2006] *bike*(vélo) contient des images de vélos sous des angles variés, à toutes les échelles et avec des fonds variés. Les images de cette base présentent de plus de nombreuses occlusions. Cette base représente bien la complexité de l'environnement des *Talking Robots*. La base contient aussi un jeu d'images de fond.

Nous avons expérimenté dans trois conditions différentes, toujours sur une tâche de classification binaire vélo/fond :

Notre premier test est un apprentissage supervisé sur l'intégralité de l'image : 150 images de la catégorie "vélo" sont utilisées pour apprendre un unique modèle d'objet. Ce modèle est ensuite testé sur 150 images de vélo et 150 images de fond.

Notre second test est similaire, mais en utilisant des fenêtres de taille 160x120 extraites de chaque image : un masque de vérité terrain fourni avec la base nous sert à déterminer la classe d'appartenance de chaque fenêtre : vélo si plus de 10% de la surface de la fenêtre est sous le masque, fond sinon. Là encore, un unique modèle est appris à partir de toutes les fenêtres de la classe vélo extraites de 150 images, puis testé sur les fenêtres de 150 autres images de la classe vélo et de 150 images de fond.

Notre troisième test est semi-supervisé : les fenêtres de taille 160x120 extraites de 50 images de vélo sont extraites et passées dans un ordre aléatoire au système dans son mode de création de modèles non supervisé, avec un taux d'acceptation $K = 1.6$ et une limite de 100 modèles. Cette valeur de K a été déterminée expérimentalement pour obtenir des modèles composés presque exclusivement d'images de fond ou d'images de vélo. Les 100 modèles obtenus

sont triés en "vélo" et "fond" en fonction de la catégorie ayant contribué le plus à sa création. Puis seuls les modèles de la classe vélo sont gardés et testés sur les fenêtres issues de 150 images de chaque catégorie. Le meilleur score obtenu sur les modèles est utilisé pour l'évaluation. La figure 5.5 montre des exemples des classes créées par le système.

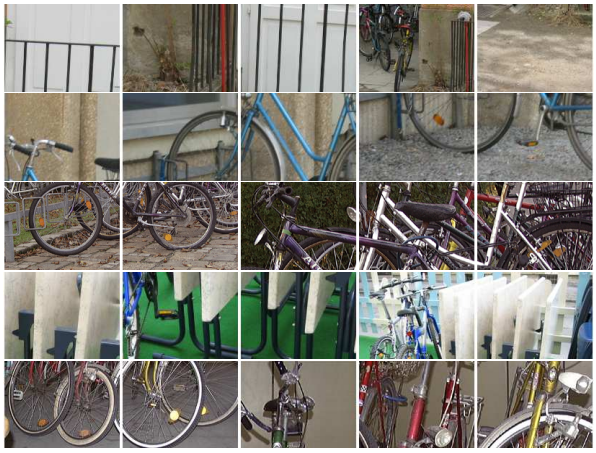


Figure 5.5: Images de 5 modèles d'objet générés de manière non supervisée.

Pour évaluer les résultats, nous utilisons les courbes ROC (*Receiver Operating Characteristic*), classiquement utilisées pour évaluer les classifieurs binaires : le classifieur associant un score à chaque image qui lui est présenté, l'utilisateur doit fixer un seuil au-dessus duquel l'image est acceptée comme faisant partie de la classe. La courbe ROC représente le taux de vrais positifs en fonction du taux de faux positifs. Elle est obtenue en faisant prendre à ce seuil toutes les valeurs possibles. Ces courbes sont souvent réduites à leur valeur au point EER (*equal error rate*), défini comme le point pour lequel la somme des vrais et des faux positifs vaut 1.

Les résultats que nous avons obtenus sont résumés figure 5.6.

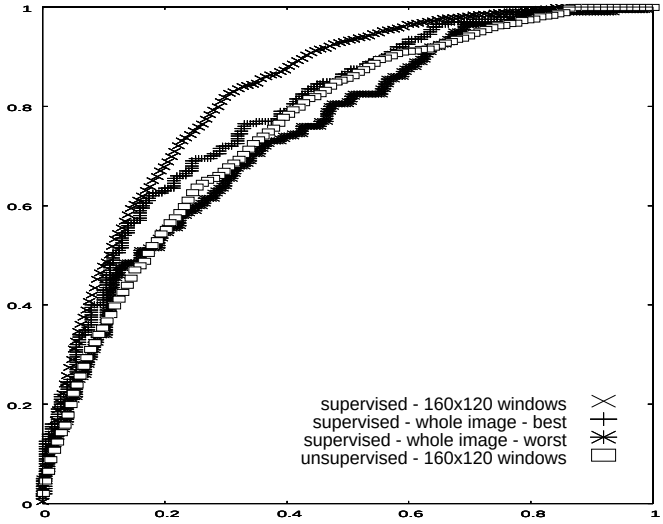


Figure 5.6: Courbes ROC sur GRAZ-02 bike.

Les résultats sur les images entières présentent une variance importante : le point ROC-EER varie entre 0.65 et 0.7 sur les dix itérations du test, ce qui signifie que les performances dépendent de manière importante des images utilisées pour l'apprentissage. Ce résultat s'explique par la grande disparité des images de la base GRAZ-02 et par le fait que les vélos n'occupent qu'une petite portion de la surface de ces images.

En utilisant les fenêtre de 160x120, le ROC-EER atteint 0.75 avec une variance beaucoup plus faible. Le test semi-supervisé atteint un ROC-EER de 0.69 en moyenne.

Ces résultats montrent que notre système peut être utilisé pour des tâches de classification non triviales. Nos performances sont en dessous de l'état de l'art sur la base GRAZ-02 (voir par exemple [Moosmann et al., 2006a]), mais notre modèle est construit avec plus de contraintes : il est complètement incrémental (le dictionnaire de caractéristiques peut être étendu, les classes peuvent être modifiées et de nouvelles classes peuvent être rajoutées) et peut fonctionner de manière non supervisée.

5.10.3 Persistance des modèles d'objets

Comme indiqué en 5.6, les *Talking Robots* vont associer des mots aux modèles d'objets. Or notre mécanisme d'apprentissage d'objets est en per-

manence en train de supprimer des modèles et d'en créer de nouveaux. Il nous faut donc vérifier que notre système est capable de conserver une partie de ses modèles à travers le temps. Pour ce faire, nous avons placé un aibo dans notre laboratoire. Celui-ci se déplace aléatoirement et fournit des images à notre système d'apprentissage. La figure 5.7 affiche à différents instants le score de chaque modèle (fréquence à laquelle il est détecté) en fonction de son rang de création. Aucun point n'est affiché pour les modèles qui ont été détruits.

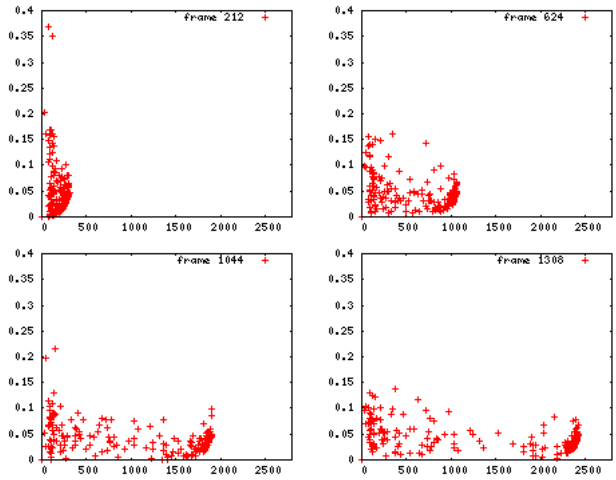


Figure 5.7: Score des modèles en fonction de leur rang de création.

On observe comme attendu un front d'objets à la droite de chaque graphe, qui correspond aux objets créés en permanence, dont la majeure partie est composée de bruit et ne sera jamais revue avant d'être supprimée une fois la période de sursis écoulée. On observe aussi qu'au bout de 1300 images, de nombreux modèles créés dans les 250 premières images sont toujours présents, ce qui montre la capacité du système à préserver des modèles fréquemment revus malgré un bruit important.

Néanmoins, notre système tel quel ne peut garder un modèle indéfiniment, car il arrivera statistiquement toujours une longue période pendant laquelle chaque modèle ne sera pas détecté et à l'issue de laquelle il sera supprimé. Une solution serait d'empêcher la suppression des modèles présents dans le lexique de l'agent avec un poids important. A défaut d'un tel mécanisme, nous avons choisi de figer les modèles d'objets pendant le déroulement du *guessing game*.

Figure 5.8: Exemples d'images prises par l'aibo dans notre laboratoire.

5.10.4 Apprentissage supervisé d'associations entre sons et objets

Avant le *guessing game*, nous avons souhaité tester notre système de reconnaissance dans un cadre supervisé avec des objets de l'environnement des *Talking Robots*. Ce test reprend de nombreux aspects et composants des *Talking Robots*, nous permettant ainsi de tester aussi la validité de notre implantation.

L'objectif de cette expérience est d'apprendre à un robot de manière supervisée et incrémentale des associations entre des objets et des sons.

Sons. Les sons, qui servent de mots dans cette expérience, sont des notes jouées sur un piano virtuel midi, sur un PC dont le haut-parleur est positionné proche de l'aibo. L'aibo réduit ce son à la composante de sa FFT de plus forte amplitude, sur une fenêtre de 20 millisecondes (cf. 4.2.1).

Objets. Nous avons choisis 7 petits objets pouvant tenir dans la main, représentés figure 5.9. Pour en faciliter la reconnaissance, l'expérimentateur agite l'objet à reconnaître devant l'aibo, qui construit un masque binaire des zones en mouvement et ignore les caractéristiques hors de ce masque.

Déroulement. L'apprentissage et l'évaluation sont une seule et même phase se déroulant de manière très similaire au *guessing game*, l'expérimentateur tenant le rôle du *speaker*.

L'expérimentateur fixe *a priori* le lexique (correspondance entre objets et son) qu'il souhaite apprendre à l'aibo.

À chaque itération, l'expérimentateur choisit un objet aléatoirement et l'agite devant la caméra de l'aibo. Le robot signale l'expérimentateur par un

beep lorsqu'il obtient un masque binaire stable, c'est-à-dire dont le centre et l'aire varient de moins d'un seuil fixé entre deux détections successives. L'expérimentateur maintient alors l'objet fixe devant la caméra : les temps d'exposition de la caméra de l'aibo sont en effet trop importants pour permettre d'exploiter une image d'un objet en mouvement.

L'Aibo cherche ensuite son modèle d'objet le plus proche de la portion de l'image sous le masque et joue le son de son lexique ayant la plus forte association. Si le son est correct, l'expérimentateur signale le succès de l'itération à l'aibo, en touchant son menton. Sinon, il signale l'échec, en touchant sa tête, et joue sur le PC la note "correcte". Dans ce cas, l'aibo rajoute la nouvelle image à l'unique modèle associé à cette note, ou en crée un nouveau à partir de cette image s'il entend la note pour la première fois.

Une vidéo d'une itération de cette expérience est disponible à l'adresse "http ://www.youtube.com/watch ?v=2_NsDmjKZ4U" .

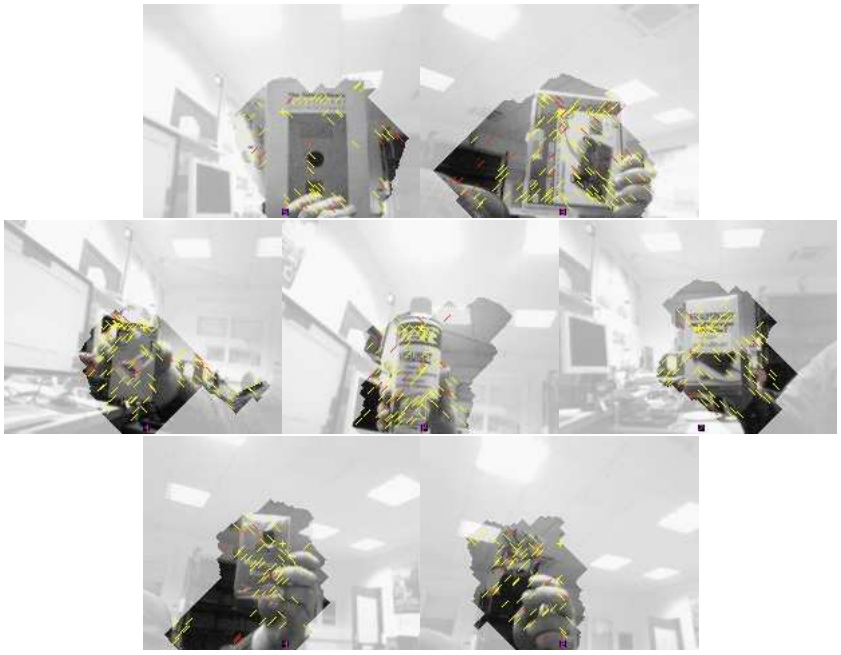


FIG. 5.9 – Objets : La partie masquée par la détection de mouvement apparaît en plus clair. Les marques jaunes marquent les positions des caractéristiques SIFT détectées.

Résultats. Les résultats sous la forme du taux moyen de succès du *guessing game*, qui correspond au taux de reconnaissance, sur 10 itérations au cours du temps sont représentés figure 5.10. On observe que ce taux se stabilise autour de 0.8 après une centaine d'itérations.

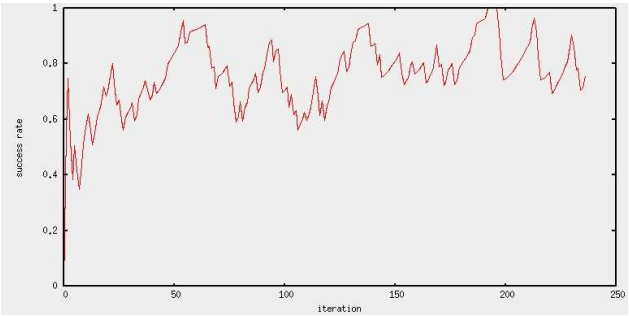


FIG. 5.10 – Résultats de la reconnaissance d'objets supervisée.

5.10.5 Limites du modèle : suppression de fond

Deux expériences montrent les limites de notre modèle concernant sa capacité à séparer un objet de son fond.

Tout d'abord, les modèles d'objets appris dans l'expérience décrite dans la section précédente ne sont pas capables de reconnaître les objets sans le masque obtenu par le mouvement si les objets sont posés sur un fond texturé. Le fond rajoute un nombre de descripteurs et donc un bruit trop important pour que la distance du modèle correct à l'image soit significativement plus faible que celle de l'image aux autres modèles.

Nous avons testé notre système dans son mode non-supervisé sur la base de données d'objets du CSCLAB (http ://csclab.ucsd.edu/labeledimages.php), utilisée par exemple dans [Murphy-Chutorian et al., 2005]. Cette base contient des images de 50 petits objets sur 10 fonds différents, seul ou par groupe de 3 à 7. Notre système crée bien des modèles correspondant aux différentes vue d'un couple (objet, fond), mais aucun modèle d'un objet sous différents fonds n'émerge. Nous avons modifié le mécanisme d'agrégation d'une nouvelle image à un modèle existant pour qu'il crée en plus un nouveau modèle "intersection", formé de l'intersection du modèle et de la nouvelle image, mais ce changement n'apporte pas d'amélioration notable.

Si notre système est bien capable de construire de manière non supervisée des classes pertinentes pour les images qui lui sont présentées, il ne s'avère

pas capable d'extraire des modèles correspondant à des sous-parties de ces images. Cette capacité n'est néanmoins pas nécessaire pour les *Talking Robots* dans un premier temps : la capacité à reconnaître de manière fiable une zone de l'environnement suffit à ses agents pour échanger des mots s'y référant.

5.10.6 *Guessing Game*

Du fait du temps nécessaire au *guessing game* (environ une minute par itération) et de l'impossibilité de faire cette expérience hors ligne sur un simulateur ou une base de donnée d'images (à cause du pointage), nous avons dû nous limiter à 1500 itérations du jeu de langage.

Nous nous sommes limité à deux agents incarnés chacun dans un aibo. Chaque agent initialise indépendamment son arbre de discrimination pour le détecteur SIFT à partir d'images prises depuis des positions aléatoires, jusqu'à atteindre environ 1000 nœuds. Puis chaque agent initialise ses modèles d'objets à partir des 16 fenêtres d'une de ces images et fournit des images au système, toujours à partir de positions et d'orientations de la tête aléatoires, jusqu'à ce que la limite des 150 modèles d'objets soit atteinte. Les modèles sont alors figés.

Les deux robots effectuent ensuite 1000 itérations du *guessing game*, un des robots jouant toujours le rôle de *speaker*, en changeant de position toutes les 10 itérations. Puis les robots effectuent 500 itérations en inversant les rôles de *hearer* et *speaker*.

A l'issue des 1500 itérations, le taux de succès du *guessing game* se situe autour de 20%. Ce taux très faible s'explique en partie par de nombreux problèmes de pointage, le spot laser tombant fréquemment sur une zone trop sombre, ou visible uniquement du *speaker*. En ignorant les échecs dues à des problèmes de pointage, ce taux de succès atteint 50%. A titre de comparaison une réponse au hasard du *hearer* donnerait un taux de succès compris entre 6% et 25%, le *speaker* choisissant comme sujet un objet apparaissant au plus sur 4 des 16 fenêtres de l'image.

Contrairement aux *Talking Heads* et à notre première version des *Talking Robots*, nous ne pouvons pas comparer directement les lexiques des deux agents. En effet, les signifiants associés aux mots dans le lexique, qui sont les modèles d'objets, sont propres à chaque agent, alors qu'ils étaient partagés pour les *Talking Heads*. Nous pouvons néanmoins observer leur structure à l'aide de la représentation de la figure 5.11.

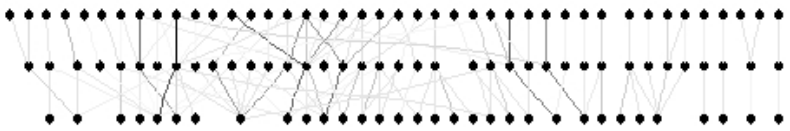


FIG. 5.11 – Lexiques des deux agents. La première ligne représente les modèles du premier agent, la deuxième les mots du lexique et la troisième les modèles du second agent. La couleur du trait donne le poids de l'association.

Les points de la ligne du milieu sont les mots, partagés par les deux agents. Les points des lignes du haut et du bas sont les modèles d'objets de chacun des agents. Le niveau de gris de chaque trait représente le poids de l'association dans le lexique.

On observe une confusion importante avec des poids faibles, au-dessus de laquelle émergent quelques associations fortes entre modèles des deux agents. L'émergence de ces quelques associations illustre le succès de la communication entre les deux agents et le partage que ce succès implique : chacun des deux agents a construit indépendamment un modèle d'une même zone de l'environnement et ces deux agents à l'issue d'interactions ont associé leurs modèles avec un même mot. L'interaction au moyen de laquelle ces associations évoluent, le *guessing game*, est conçue de manière à nous garantir leur aspect "opérationnel" : les mots produits pourront être utilisés lors de communications à but utile par les agents pour se désigner des régions de leur environnement.

5.11 Différence avec les Talking Heads

Dans notre recherche d'un système capable de reproduire les résultats des *Talking Heads* dans un environnement plus riche, nous nous sommes éloigné de l'expérience initiale sur deux points importants.

Niveau d'abstraction des signifiants. Les signifiants des *Talking Heads* sont les nœuds de ses arbres de discrimination et correspondent à des propriétés abstraites des objets auxquels ils se réfèrent comme par exemple leur taille, ou leur couleur. Les signifiants des *Talking Robots* sont les modèles d'objets eux-mêmes. En conséquence, les mots des *Talking Heads* paraissent plus facilement réutilisables dans un environnement nouveau inconnu des agents. Cette différence est causée par une simplification importante du

modèle d'environnement des *Talking Heads* par rapport à la réalité : les objets sont donnés et délimités *a priori*. Dans notre modèle plus proche de la réalité dans lequel les robots doivent apprendre les limites des objets, les agents ne peuvent communiquer sur des propriétés abstraites des objets avant de pouvoir communiquer sur les objets eux-mêmes. Ce modèle semble correspondre aux observations de la psychologie développementale [Smith et al., 1999], qui tendent à montrer que les bébés apprennent d'abord les concepts se référant à des objets entiers, puis les concepts se référant à des propriétés ou parties de ces objets.

Salience. Les *Talking Heads* utilisent un mécanisme de salience visuelle, pour éliminer *a priori* du contexte les objets que les agents ne sont pas capables de discriminer de manière robuste (c.f. 2.3) et pour tenter de lever les ambiguïtés lors de l'apprentissage. Ce mécanisme est complètement absent des *Talking Robots*. Concernant le deuxième usage, les *Talking Robots* n'ont pas de problème d'ambiguïté au moment de l'apprentissage, car c'est tout l'objet qui est appris et non une de ses propriétés que l'agent doit sélectionner. Concernant le filtrage des objets du contexte, nous avons expérimenté sans succès avec des algorithmes de salience visuelle, du type [Itti et al., 1998] : les N zones les plus salientes données par ces algorithmes ne sont pas suffisamment robustes à de légers changements de point de vue. À noter que les mêmes auteurs dans [Carmi and Itti, 2006] montrent que les modèles dynamiques utilisant l'information de mouvement sont de bien meilleurs prédicteurs des saccades de l'oeil (et donc de meilleurs modèles de la salience perçue par les Humains) que les modèles statiques. Nos agents observant des scènes statiques, cette forme de salience, qui serait plus robuste, n'est pas disponible. De nombreux résultats de psychologie développementale confirment l'importance de la salience dans l'apprentissage de leurs premiers mots par les bébés, mais cette salience est principalement basée sur le mouvement (voir par exemple [Matatyaho and Gogate, 2006]).

5.12 Futurs développements

Comme nous l'avons montré en 5.10.6, les *Talking Robots* sont capables de partager un même signifiant pour une portion des mots de leur vocabulaire, mais la confusion reste importante, le taux d'erreur du *guessing game* étant de l'ordre de 50%.

De nombreuses améliorations peuvent être apportées pour réduire cette confusion. La première consisterait à remplacer le système de fenêtrage, qui impose de rechercher des objets à des positions arbitraires dans l'image, par un système plus adaptatif, s'adaptant à la position des caractéristiques détectées. Une implantation possible serait de conserver les fenêtres superposées, mais d'utiliser un mécanisme d'inhibition latérale pour les mettre en compétition pour la possession des caractéristiques. Le système ne détecterait ainsi plus nécessairement un objet sur chaque fenêtre, ce qui limiterait considérablement la quantité de modèles "bruit" générés.

Nous envisageons aussi de réintroduire l'information de position relative entre caractéristiques, absente de notre modèle "sac de mots", par le biais d'un nouveau détecteur de caractéristiques se basant sur les autres détecteurs. Sa sortie serait un couple formé d'un sous ensemble des caractéristiques et d'une représentation de leur relation spatiale. L'information d'orientation fournie par certains détecteurs comme SIFT peut aussi être utilisée (voir 5.2 pour une rapide analyse de la littérature utilisant ces méthodes).

Si après ces modifications le système atteint un niveau de performance satisfaisant, plusieurs voies d'explorations sont possibles.

Tout d'abord, il serait intéressant d'appliquer un processus sélectionniste à la génération des mots, qui pour le moment sont générés aléatoirement à partir d'une liste de syllabes et échangés à travers le réseau : chaque agent disposerait d'un processus pour générer des sons, à partir par exemple d'un modèle simple d'appareil vocal, et d'algorithmes de segmentation et de classification sonores, qui seraient utilisés pour créer des modèles de son, de manière similaire aux modèles d'objets des *Talking Robots* ([Oudeyer, 2005]). Ainsi, tous les éléments de la mémoire des agents (modèles de mot, modèles d'objets) seraient appris. Le résultat serait un modèle de communication entre agents, sur et à travers leur environnement, construit incrémentalement, de manière non supervisée et sans aucun *a priori* autre que des capacités motrices et perceptives identiques pour ces agents.

L'étape suivante consisterait alors à ne plus coder en dur la logique du *guessing game* dans les agents, mais à les laisser se développer leurs propres interactions, à l'aide d'un mécanisme sélectionniste de génération de comportements privilégiant ceux faisant "progresser" l'agent. Le choix de la grammaire de comportements et du critère d'utilité des comportements générés sera crucial, d'autant plus que les comportements appliqués par deux agents en interaction doivent se correspondre pour pouvoir être utiles. Le lecteur intéressé pourra se reporter à [Baillie and Nottale, 2005a] pour une

analyse plus poussée de la question.

Enfin, même si un taux de succès élevé du *guessing game* est suffisant pour démontrer le caractère "utile" des mots échangés par les agents, nous aimerions intégrer cette expérience dans une simulation plus vaste, dans laquelle les agents auraient à accomplir des tâches qui se trouveraient facilitées par une action coordonnée et une communication efficace.

Chapitre 6

Conclusions

Cette thèse a pour objectif d'étendre les *Talking Heads* à un environnement perceptuel plus riche en utilisant des robots mobiles. L'intérêt de cette démarche est double : vérifier dans un premier temps la pertinence du modèle des *Talking Heads* dans un cadre plus complexe, puis profiter de cette complexité nouvelle pour étendre les *Talking Heads* en augmentant progressivement leur autonomie. La relative simplicité de l'environnement initial des *Talking Heads* pose en effet une limite à leur développement.

Les récentes avancées dans les domaines de la robotique (disponibilité de plates-formes mobiles dotés de caméra à bas coût) et de la vision artificielle (algorithmes de détection de points d'intérêt) rendent de plus le moment propice pour un tel développement.

Nous avons dans un premier temps tenté une implantation restant aussi proche que possible de la version initiale des *Talking Heads*, en utilisant un algorithme de segmentation d'image pour retrouver un environnement conceptuellement identique, c'est-à-dire composé d'un ensemble d'objets correspondant chacun à une zone de l'image, associés à des propriétés pour un certain nombre de canaux perceptifs prédéfinis. Nous n'avons pas réussi à obtenir convergence des lexiques des agents en utilisant ce modèle : les perceptions des deux agents diffèrent de manière trop importante et, en l'absence d'un environnement partagé, aucune communication n'est possible. Nous avons expérimenté avec diverses méthodes de stabilisation des segmentations sans amélioration notable des résultats.

Partant du constat qu'aucun algorithme existant ne pouvait nous donner de manière robuste une notion d'objet à partir d'une seule image, nous avons développé un modèle capable de construire cette notion de manière

incrémentale à partir de l'existant dans le domaine de la reconnaissance d'objets. Notre système définit un modèle d'objet comme le vecteur d'occurrences de mots d'un dictionnaire de descripteurs en présence de l'objet. Il peut construire de manière incrémentale et non supervisée à la fois le dictionnaire de descripteurs et les modèles d'objets, en utilisant un critère de décision simple pour choisir entre intégrer une image au modèle le plus proche, ou créer un nouveau modèle à partir de cette image.

Des expériences préliminaires montrent sa capacité à classifier sur des bases de données d'images et à conserver au cours du temps des modèles d'objets stables malgré un bruit important.

Les expériences de *guessing game* montrent malgré un taux de succès global qui reste faible (mais significatif) l'émergence d'associations fortes entre les modèles des deux agents à travers leur lexique, montrant qu'une communication utilisant des symboles ancrés dans leur environnement et désignant des zones de celui-ci a effectivement lieu entre ces agents.

Ainsi, même si un travail important reste à accomplir pour rendre le processus de création de modèles d'objets plus robuste, notre système constitue une bonne base de départ pour étendre les *Talking Robots* vers des interactions plus complexes.

Annexe A

Génération de graphes de comportement en C++

La lecture des valeurs capteurs de l'aibo (caméra, micro, position des servomoteurs) et l'envoi de commandes se font en utilisant le langage Urbi, langage initialement développée à l'ensta, puis par la société Gostai. Toutes les opérations de traitement d'image et la logique des *Talking Robots* se font sur un PC distant, relié au robot par TCP/IP. A l'époque où ces expériences ont été réalisées, le langage Urbi n'était pas assez avancé pour exprimer la logique de fonctionnement des *Talking Robots*. Il nous a donc fallu développer un système capable d'isoler et d'exprimer simplement un graphe de comportements en C++ et permettant de s'abstraire du mécanisme asynchrone sous-jacent d'acquisition de données.

Notre système s'inspire des modules spirit : :phoenix et lamba de la bibliothèque boost.

Son premier module offre un moyen de créer et de manipuler des expressions en C++ : Il définit tout d'abord un type abstrait *Value*, ayant la capacité à s'évaluer pour retourner une valeur. Puis il définit 3 sous-classes 'primitives' de Value : constante, variable C++ et variable Urbi qui, lorsqu'elles sont évaluées, retournent respectivement une constante, la valeur d'une variable C++ ou la valeur d'une variable Urbi (pouvant être un capteur du robot).

Il définit ensuite tous les opérateurs standards sur les *Value* (+, -,*, /, min et max, sqrt...).

Il définit ensuite un type Variable, pouvant être une variable C++ ou une variable Urbi, et un opérateur << permettant d'affecter une Valeur à une

Variable.

Son deuxième module permet d'exprimer un graphe à partir de nœuds élémentaires et de transitions construites à base de *Value* et de *Variable*.

Chaque nœud élémentaire est une classe implantant l'interface Node (3 méthodes *enter*, *run* et *leave* appelées par le mécanisme d'exécution de graphes) et réalise une des tâches élémentaires dont nous avons besoin : émettre un son, déplacer un moteur, rechercher un spot laser, segmenter la scène... Chaque nœud prend ses paramètres sous la forme de *Value* et les évalue a chaque fois qu'il en à besoin.

Un graphe est constitué en enchaînant des nœuds et des *Value*, qui servent de transition.

Un exemple permettra d'éclaircir cette description : supposons que nous disposions d'une classe nœud *FindBall*, prenant en paramètre trois variables dans lesquelles elle écrit la position en x, la position en y, et 1 si la balle rose de l'aibo est visible, respectivement, et d'une classe nœud Move, prenant en paramètre un nom de moteur d'aibo et une *Value*. On peut écrire le code suivant :

```
 1 :  #include "graph.hh"
 2 :  float balllx, bally;
 3 :  bool ballVisible;
 4 :
 5 :  int main() {
 6 :    Variable ballx_(ballx), bally_(bally), ballVisible_(ballVisible)
 7 :      Node &fb = findBall(ballx_ ,bally_, ballVisible_);
 8 :
 9 :    Node end;
10 :
11 :    fb >> (ballVisible == Value(true)) >> move("headPan",
12 :        ballx_*50.0/208) >> move("headTilt", ballly_*40/160) >> fb;
13 :
14 :    fb >> timeout(100000) >> move("headPan", 50-random(100))
15 :      >> fb;
16 :
17 :    fb >> (UVariable("headSensor") > 0) >> end;
18 :    Graph g(fb);
19 :    g.setEndState(end);
20 :    g.execute();
21 :  }
```

Ce graphe, lorsqu'il est exécuté, fait chercher au robot la balle et tourner la tête dans sa direction une fois qu'il la voit. Si la balle n'est pas trouvée au bout de 10 secondes, le robot tourne sa tête dans une direction aléatoire et recommence.

Le graphe s'arrête lorsqu'on appuie sur la tête du robot.

Le symbole >> est utilisé pour lier une transition à deux états, ou deux états ensemble par une transition ϵ.

Nous nous heurtons au même problème que tous les systèmes similaires : pour qu'une expression soit reconnue comme *Value*, il faut qu'au moins un de ses membres soit de type *Value*, sinon l'expression est évaluée normalement au moment ou le code C++ est exécuté. Cette contrainte alourdit par moment la syntaxe, en forçant à tiliser des transtypages explicites.

Nous avons aussi implanté un opérateur || entre deux nœuds, permettant de créer un nouveau nœud qui exécutera en parallèle ses deux opérandes.

Ce système, complexe dans son implantation, permet d'exprimer simplement un graphe de comportement à partir de briques élémentaires en C++, et donc d'accélérer le développement d'expériences de robotique.

Bibliographie

John R. Anderson and Christian Lebiere. *The atomic components of thought.* Lawrence Erlbaum, 1998.

Jean-Christophe Baillie. Urbi : Towards a universal robotic low-level programming language. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005.

Jean-Christophe Baillie and Matthieu Nottale. Dynamic evolution of language games between two autonomous robots. In *Proceedings of the IEEE International Conference on Development and Learning*, 2005a.

Jean-Christophe Baillie and Matthieu Nottale. Segmentation stability : A key component for joint attention. In *5th International Conference on Epigenetic Robotics*, 2005b.

R.A. Boie and I.J. Cox. An analysis of camera noise. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(6) :671–674, 1992.

Rodney A. Brooks. Elephants don't play chess. *robotics and autonomous systems*, 6, 1990.

Rodney A. Brooks. A robust layered control system for a mobile robot. *IEEE journal of robotics and automation*, RA-2(1) :14– 23, march 1986.

Ran Carmi and Laurent Itti. Causal saliency effects during natural vision. In *Proceedings of the 2006 symposium on Eye tracking research & applications*, pages 11–18, 2006.

Qin-Sheng Chen, Michel Defrise, and F. Deconinck. Symmetric phase-only matched filtering of fourier-mellin transforms for image registration and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(12) :1156–1168, 1994.

D. Comaniciu and P. Meer. Robust analysis of feature spaces : color image segmentation. In *CVPR '97 : Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, page 700. IEEE Computer Society, 1997. ISBN 0-8186-7822-4.

P.L. Correia and F. Pereira. Objective evaluation of video segmentation quality. *IEEE Transactions on Image Processing*, 12(2) :186 – 200, 2003.

Chris Dance, Jutta Willamowski, Lixin Fan, Cedric Bray, and Gabriela Csurka. Visual categorization with bags of keypoints. In *ECCV International Workshop on Statistical Learning in Computer Vision, Prague*, 2004.

Andrew Davison. Real-time simultaneous localisation and mapping with a single camera. In *IEEE International Conference on Computer Vision*, pages 1403–1410, October 2003. ISBN 0-7695-1950-4. URL `http ://pubs.doc.ic.ac.uk/single-camera-slam/`.

de Jong E. The development of a lexicon based on behavior. In *Proceedings of the Tenth Netherlands /Belgium Conference on Artificial Intelligence*, pages 27–36, 1998.

S. Doncieux and J.-A. Meyer. L'approche animat et la robotique évolutionniste. In *JNRR'03 - Quatrièmes Journées Nationales de Recherche en Robotique.*, 2003.

L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples : an incremental bayesian approach tested on 101 object categories. In *IEEE. CVPR 2004, Workshop on Generative-Model Based Vision.*, 2004.

Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE Pattern Analysis and Machine Intelligence*, 28(4) :594–611, April 2006.

Vittorio Ferrari, Loic Fevrier, Frederic Jurie, and Cordelia Schmid. Groups of adjacent contour segments for object detection. INRIA technical report, august 2006.

D. Filliat. A visual bag of words method for interactive qualitative localization and mapping. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 3921–3926, 2007.

Stevan Harnad. The symbol grounding problem. *Physica D*, 42 :335–346, 1990.

C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988.

Eric Hjelmas and Boon Kee Low. Face detection : A survey. *Computer Vision and Image Understanding*, 83 :236–274, 2001.

J. Huang, S. Kumar, M. Mitra, W. Zhu, and R. Zabih. Image indexing using color correlograms. In *IEEE Computer Vision and Pattern Recognition*, pages 762–768, 1997. URL `citeseer.ist.psu.edu/huang97image.html`.

Qian Huang and B. Dom. Quantitative methods of evaluating image segmentation. In *Proceedings of the International Conference on Image Processing*, volume 3, pages 53–56, 1995.

Erik Murphy-Chutorian Huei-Ju Chen, Kuang-Chih Lee and Jochen Triesch. *Advances in Visual Computing*, chapter Toward a Unified Probabilistic Framework for Object Recognition and Segmentation, pages 108–117. Springer Berlin / Heidelberg, 2005.

Takashi Ikegami and Hiroki Iizuka. Simulated turn-taking and development of styles of motion. Imitation and Social Learning in Robots, Humans and Animals, Chapter 14. Cambrige University Press, ISBN 9780521845113, March 2007.

L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Pattern Analysis and Machine Intelligence*, 20(11) :1254–1259, 1998.

S. Jodogne, F. Scalzo, and J.H. Piater. Task-driven learning of spatial combinations of visual features. In *Computer Vision and Pattern Recognition (CVPR)*, pages 48 – 48, 2005.

Frederic Jurie and Bill Triggs. Creating efficient codebooks for visual recognition. In *International Conference on Computer Vision*, volume 1, pages 604–610, 2005.

Frederic Kaplan. Talking AIBO : First experimentation of verbal interactions with an autonomous four-legged robot. In *Proceedings of the CELE-Twente workshop on interacting agents*, 2000.

John E. Laird and Paul S. Rosenbloom. The evolution of the soar architecture. 1994.

S Lazebnik, C Schmid, and J Ponce. Beyond bags of features : Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition*, volume 2, pages 2169– 2178, 2006.

Thomas M. Lehmann. A two-stage algorithm for model-based registration of medical images. In *Proceedings of the 14th International Conference on Pattern Recognition-Volume 1*, page 344. IEEE Computer Society, 1998. ISBN 0-8186-8512-3.

Haibin Ling and Kazunori Okada. Diffusion distance for histogram comparison. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 246–253, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2597-0. doi : http ://dx.doi.org/10.1109/CVPR.2006.99.

D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal on Computer Vision*, 60(2) :91–110, 2004.

A. Manzanera. Morphological segmentation on the programmable retina : towards mixed synchronous/asynchronous algorithms. In *Proceedings of 6th International Symposium on Mathematical Morphology (ISMM'02), Sydney, Australia, April 2002*, pages 389–399, 2002.

Dalit J. Matatyaho and Lakshmi J. Gogate. The role of motion in maternal bimodal naming to preverbal infants. In *International conference on developmental learning*, 2006.

John McCarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969. reprinted in McC90.

Lewin Michael. Concept formation and language sharing : Combining steels' language games with simple competitive learning. Master's thesis, University of Sussex, 2002.

Marvin Minsky. Steps towards artificial intelligence. Technical report, Dept. of Mathematics, MIT, 1960.

Marvin Minsky. Why people think computers can't. *AI Magazine*, 3(4), 1982.

Frank Moosmann, Diane Larlus, and Frederic Jurie. Learning saliency maps for object categorization. In *ECCV International Workshop on The Representation and Use of Prior Knowledge in Vision*. Springer, 2006a. URL `http ://lear.inrialpes.fr/pubs/2006/MLJ06`.

Frank Moosmann, Bill Triggs, and Frédéric Jurie. Fast discriminative visual codebooks using randomized clustering forests. In *NIPS*, pages 985–992, 2006b.

Erik Murphy-Chutorian, Sarah Aboutalib, and Jochen Triesch. Analysis of a biologically-inspired system for real-time object recognition. *Cognitive Science Online*, 3.2 :1–14, 2005.

D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 2161–2168, June 2006.

Stephen M. Omohundro. Five balltree construction algorithms. Technical Report tr-89-063, International Computer Science Institute, 1989.

A. Opelt, A. Pinz, M. Fussenegger, and P. Auer. Generic object recognition with boosting. *IEEE Transactions on Pattern Recognition and Machine Intelligence (PAMI)*, 28(3) :416–431, 2006.

P-Y Oudeyer. How phonological structures can be culturally selected for learnability. *Adaptive Behavior*, 13(4) :269–280, 2005.

Pierre-Yves Oudeyer, Frederic Kaplan, and Verena V. Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, 11(2) :265–286, april 2007.

Lutz Priese and Volker Rehrmann. A fast hybrid color segmentation method. In *DAGM-Symposium*, pages 297–304, 1993.

T. Quack, V. Ferrari, and L.J. Van Gool. Video mining with frequent itemset configuration. In *International Conference on Image and Video Retreival (CIVR)*, pages 360–369, 2006.

F Rosenblatt. The perceptron : A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65 :386–408, November 1958.

Y. Rubner, C. Tomasi, and L. J. Guibas. A metric for distributions with applications to image databases. *Proceedings of the 1998 IEEE International Conference on Computer Vision*, pages 59–66, 1998.

John Searle. Minds, brains and programs. *Behavioral and brain sciences,*, 3 :417–421, 1980.

J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman. Discovering object categories in image collections. In *International Conference on Computer Vision*, 2005.

Josev Sivic and Andrew Zisserman. Video google : A text retrieval approach to object matching in videos. In *Ninth IEEE International Conference on Computer Vision*, volume 2, pages 1470– 1477, 2003.

L.B. Smith, M. Gasser, and C. M. Sandhofer. *Perceptual learning. The psychology of learning and motivation,*, volume 36, chapter Learning to talk about the properties of objects : A network model of the development of dimensions., pages 219–255. US : Academic Press., 1999.

D. Squire, W. Muller, H. Muller, and J. Raki. Content-based query of image databases, inspirations from text retrieval : inverted files, frequency-based weights and relevance feedback. The 10th Scandinavian Conference on Image Analysis (SCIA'99), 1999.

L. Steels. The symbol grounding problem is solved, so what's next ? In M. De Vega, G. Glennberg, and G. Graesser, editors, *Symbols, embodiment and meaning*. Academic Press, New Haven, 2007. URL `http ://www.isrl.uiuc.edu/ amag/langev/paper/steels07symbolGrounding.html`.

Luc Steels. *The talking heads experiment*. LABORATORIUM,antwerpen, 1999.

Luc Steels. The origin of syntax in visually grounded robotic agents. *Artificial Intelligence*, 103 :133–156, 1998.

Luc Steels and Jean-Christophe Baillie. Shared grounding of event descriptions by autonomous robots. *Robotics and autonomous systems*, 43 :163–173, 2003.

M. Taddeo and L. Floridi. Solving the symbol grounding problem : a critical review of fifteen years of research. *Journal of Experimental and Theoretical Artificial Intelligence*, 17 :419–445, 2005. URL `http ://philsci-archive.pitt.edu/archive/00002542/`.

Paul Vogt. The emergence of compositional structures in perceptualy grounded language games. *Artificial intelligence*, special issue on connecting languages to the world, 2005.

Paul Vogt. *Lexicon Grounding on Mobile Robots*. PhD thesis, Vrije Universiteit Brussel, 2000.

Paul Vogt. Minimum cost and the emergence of the zipf-mandelbrot law. In *Artificial Life IX : Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems*, 2004.

Paul Vogt and Hans Coumans. Investigating social interaction strategies for bootstrapping lexicon development. *Journal of Artificial Societies and Social Simulation*, 6(1), 1 2003. URL `http ://www.isrl.uiuc.edu/ amag/langev/paper/vogt03jasss.html`.

Paul Vogt and F Divina. Language evolution in large populations of autonomous agents : issues in scaling. In *Proceedings of AISB 2005 : Social Intelligence and Interaction in Animals, Robots and Agents.*, 2005.

Michael Werman and Daphna Weinshall. Similarity and affine invariant distances between 2d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8) :810–814, 1995.

G. Wolberg and S. Zokai. Robust image registration using log-polar transform. *IEEE Proceedings of International Conference on Image Processing*, 1 :493–496, 2000. URL `citeseer.ist.psu.edu/wolberg00robust.html`.

Barbara Zitova and Jan Flusser. Image registration methods : a survey. *Image and Vision Computing*, 21 :977–1000, 2003.