Institut **S**upérieur
d'**I**nformatique de
Modélisation et de
leurs **A**pplications

Complexe des Cézeaux
BP 125
63173 Aubière Cedex

Equipe **FLOWERS**
Institut **N**ational de
Recherche en
Informatique et en
Automatique

INRIA Bordeaux
351, cours de la Libération
Bâtiment A29
33405 Talence Cedex

Rapport de stage de 3ème année

ISIMA Génie Logiciel – Master Recherche Informatique option Robotique

Apprentissage supervisé avec association sons/images en robotique développementale

Présenté par : **Bérenger BRAMAS**

Responsable : **Pierre-Yves OUDEYER**

Remerciements

Je tiens à remercier tous les membres de l'équipe FLOWERS. Plus particulièrement, je tiens à remercier Pierre-Yves OUDEYER qui a accepté de m'accueillir et m'a guidé tout au long de mon stage. Ma reconnaissance va aux différents membres de l'équipe qui m'ont conseillé et aidé sur des problèmes techniques et scientifiques.

Glossaire

Apprentissage (*Machine Learning*) : C'est une discipline scientifique liée à la recherche et au développement d'algorithmes dans le but de donner les moyens aux machines d'évoluer grâce à un processus d'apprentissage. L'objectif est que les machines remplissent des tâches difficilement abordables avec la programmation classique.

Humanoïde (robot): Est un terme utilisé pour caractériser les robots de forme humaine. C'est-à-dire ceux qui possèdent deux bras, deux jambes, un buste et une tête. Le plus connu est Asimo développé par Honda.

HRI : *Human-Robot Interaction* (interactions entre humains et robots) : c'est l'étude des interactions entre les utilisateurs (humains) et les robots. En langue française, il n'existe pas d'acronyme pour désigner ce domaine. Dans ce rapport, l'acronyme anglais HRI sera utilisé pour exprimer "interaction entre robots et humains".

IA (Intelligence Artificielle) : est la « recherche de moyens susceptibles de doter les systèmes informatiques de capacités intellectuelles comparables à celles des êtres humains » (CNRTL).

Langage C++ : Langage de programmation orienté objet mis au point par Bjarne Stroustrup au début des années 1980. Le C++ est un langage puissant très utilisé dans l'industrie informatique.

Langage orienté objet : Langage de programmation reposant sur l'assemblage de briques logicielles appelées objets.

Librairie (ou Bibliothèque) : En informatique, c'est un ensemble de fonctions utilitaires, regroupées et mises à disposition afin de pouvoir être utilisées sans avoir à les réécrire.

Phonème : Représente la plus petite unité que l'on peut isoler dans une phrase. Les phonèmes sont différents selon les langues.

QT : C'est une bibliothèque logicielle orientée objet et développée en C++ par la société Trolltech (Nokia). Elle offre des composants d'interface graphique, d'accès aux données, de connexions réseaux, de gestion des fils d'exécution, d'analyse XML, etc. et ce sur de multiples plateformes.

Reconnaissance Vocal : C'est une technologie qui permet de transcrire du son en données exploitables par la machine. Le plus souvent, le son est transcrit en texte.

Robot : C'est un dispositif mécanique accomplissant automatiquement des tâches généralement considérées comme dangereuses, pénibles ou impossibles pour les humains. La définition de ce qui est considéré ou non comme robot varie selon les pays et les cultures.

SE/OS : Système d'Exploitation/Operating System : est un ensemble de programmes responsables de la liaison entre les ressources matérielles d'un ordinateur et les applications informatiques de l'utilisateur. Exemples : Windows, Linux, Mac OS.

TTS : *Text To Speech* : Littéralement du « texte à la parole », le TTS représente tous les procédés qui permettent la génération d'une voix à partir d'un texte.

Processus : Un processus est défini par un ensemble d'instructions à exécuter (un programme), un espace mémoire pour les données de travail et éventuellement, d'autres ressources, comme des descripteurs de fichiers, des ports réseau, etc.

Processus léger/*Theads* : Est similaire à un processus car tous deux représentent l'exécution d'un ensemble d'instructions du langage machine d'un processeur. Du point de vue de l'utilisateur, ces exécutions semblent se dérouler en parallèle. Toutefois, là où chaque processus possède sa propre mémoire virtuelle, les processus légers, appartenant au même processus père, se partagent sa mémoire virtuelle.

Résumé

Les **robots** ludiques et sociaux sont amenés à interagir avec les humains. Dans ce cadre, ils doivent apporter l'illusion d'être vivant auprès des utilisateurs. Pour cela, différentes recherches sont faites dans le but de donner aux robots des comportements et la capacité d'évoluer. L'objectif est qu'ils soient capables d'acquérir de nouveaux savoir-faire, de nouvelles connaissances et d'être robuste aux variations de l'environnement. Toutefois, pour que les utilisateurs apprécient les échanges avec les robots, il faut que les interactions soient naturelles et faciles même pour un non spécialiste. Par exemple, les utilisateurs qui interagissent avec un robot attendent de celui-ci qu'il évolue et apprenne des informations sur son environnement pour le partager ensuite avec eux.

Ainsi, mon travail a consisté à effectuer des recherches dans le but de concevoir un système d'**apprentissage**. Ce système permet d'enseigner des objets à un robot en utilisant la parole et une caméra. Ce système respecte des contraintes particulières comme l'absence de pré-requis au départ et la création de catégorie au fur et à mesure de l'apprentissage. De plus, le système peut apprendre en temps réel, c'est-à-dire immédiatement et avec peu d'exemples.

La **reconnaissance vocale** repose sur des calculs répandus et très utilisés pour le traitement de la parole (PLP-Rasta/DTW). En revanche, le **traitement de l'image** a demandé un travail important pour créer des algorithmes permettant de retrouver des objets dans un environnement hostile. Ces algorithmes sont basés sur les descripteurs de type **SURF/SIFT**.

Le système permet d'enseigner des objets, de façon intuitive, en les agitant devant une caméra et en prononçant des mots. Enfin, une phase de test a été réalisée sur humanoïde pour utiliser le système en situation réelle.

Mots clés: robot, apprentissage, HRI, reconnaissance vocale, imagerie, surf/sift.

Abstract

Social **robots** lead up to interact more and more with humans. In that context, they have to give the impression of being alive. To do that, many researchers try to give robots the ability to evolve and to simulate different behaviors. The main objective is to make the robot able to gain new abilities, new knowledge or to be robust to environmental changes. However, in order for the users to appreciate interactions with robots, **interactions** have to be simple and natural even for a non specialist user. For example, users expect from a robot to evolve and learn information about world to share with them.

Thereby, my work consisted in researching and creating a **learning** system. This system enables to teach objects to a robot using voice and a camera. It respects hard constraints like learning without a priori knowledge or the creation of categories during the learning process. Moreover, it can learn in real time and with few examples.

Speech recognition is based on the very popular PLP-Rasta analysis. Then, we compare sound using the dynamic time wrapping algorithm. On the other hand, **image processing** required creating new methods to succeed recognizing objects in hostile environment. Theses algorithms use the **SURF/SIFT** invariant descriptors.

The final system enables users to teach object in a very intuitive way by moving the objects in front of the camera and pronouncing their names. Finally, the system was tested in a real use on a humanoid robot.

Keywords: Robots, learning, HRI, speech recognition, image processing, surf/sift.

Table des figures

Figure 1 : Image visible et extraction d'information possible (*back propagation,* histogramme et contour) 4
Figure 2 : Exemple IPhone 4
Figure 3 : Exemple d'évolutions possibles lors d'un blocage 5
Figure 4 : onde sonore « livre noir [silence] livre » 8
Figure 5 : Exemple PLP-Rasta 9
Figure 6 : Exemple de correspondance SIFT 10
Figure 7 : Représentation de la base de données visuelle 11
Figure 8 : Principe d'extraction du contour 12
Figure 9 : Problème de correspondance 13
Figure 10 : Exemples d'illustrations de la méthode de David LOWE 15
Figure 11 : Exemples d'illustrations par la méthode de SEUIL (0.6 et 0.3) 16
Figure 12 : Exemple d'illustration par la méthode de la meilleure correspondance possible 17
Figure 13 : Principe du filtre spatial 18
Figure 14 : Principe du filtre angulaire 19
Figure 15 : Principe de reconnaissance 21
Figure 16 : Résultats du test de reconnaissance 22
Figure 17 : Illustration des résultats de comparaisons 22
Figure 18 : Variations des résultats en fonction du seuil de correspondances 23
Figure 19 : Phase d'apprentissage 24
Figure 20 : Correspondance espaces visuel/acoustique 25
Figure 21 : Zone d'intérêt 26
Figure 22 : Structure du système 28
Figure 23 : Interfaces Matlab – Qt – OpenCv 28
Figure 24 : Nao 29
Figure 25 : Position clés de Nao pour s'exprimer 30
Figure 26 : Structure du système avec Nao et URBI 33
Figure 27 : Organisation du stage 34

Table des algorithmes et codes sources

Algorithme 1 : Méthode de comparaison proposé par David LOWE 14
Algorithme 2 : Méthode de correspondance par SEUIL 15
Algorithme 3 : Méthode de la meilleure correspondance possible 17
Algorithme 4 : Filtre spatial 18
Algorithme 5 : Meilleure image 21
Algorithme 6 : Regroupement des points en utilisant la distance moyenne du plus proche voisin 27
Algorithme 7 : Exemple de script URBI sur Nao (Code) 32

Table des matières

INTRODUCTION **1**

1 INTRODUCTION AU SUJET DE RECHERCHE **2**

1.1 Présentation du cadre **2**
1.1.1 L'INRIA 2
1.1.2 L'équipe FLOWERS 2

1.2 La robotique développementale **3**
1.2.1 Définition générale et problématique 3
1.2.2 Exemple 3

1.3 Le sujet **5**
1.3.1 Objectif 5
1.3.2 Problématiques 5
1.3.3 Travaux connexes 5

2 METHODOLOGIE **8**

2.1 Traitements sonore **8**
2.1.1 Choix et représentation 8
2.1.2 Capture du son 8
2.1.3 Traitements 8

2.2 Traitement de l'image **9**
2.2.1 SURF/SIFT 10
2.2.2 Extraction de contour 10
2.2.3 Correspondances 13
2.2.4 Filtre spatial 17
2.2.5 Filtre angulaire 18
2.2.6 Critère de choix 19

2.3 Résultats de la reconnaissance visuelle **21**

2.4 Phase d'apprentissage et association **24**
2.4.1 Déroulement de l'apprentissage 24
2.4.2 Relation sons - images 24
2.4.3 Restitution 25

2.5 Aspects techniques **28**

2.6 Utilisation sur humanoïde **29**
2.6.1 Introduction 29
2.6.2 Nao 29
2.6.3 URBI 31
2.6.4 Structure du système 32

2.7 Limites de l'architecture **33**

3 DISCUSSION ET PERSPECTIVE **34**

3.1 Organisation du stage **34**

3.2 Perspectives **35**

CONCLUSION **37**

BIBLIOGRAPHIE **38**

ANNEXES **40**

Introduction

Du 6 avril au 25 septembre 2009, j'ai effectué mon stage de dernière année d'école d'ingénieur et de master recherche au sein du de l'équipe FLOWERS (*FLOWing Epigenetic Robots and Systems*) à l'INRIA Bordeaux. Cette équipe travaille sur des domaines liés à la robotique développementale et l'IA.

Parmi ces domaines de recherche, mon travail est directement lié au développement d'interfaces intuitives et naturelles pour les utilisateurs, ainsi qu'à l'apprentissage et la perception du monde par un robot.

L'objectif de ce stage a été l'étude et la conception d'un système qui permet d'enseigner des objets de façon naturelle à un robot. Ce qui pose différents problèmes techniques et formels. J'ai du concevoir un système qui permet de montrer facilement des objets à un robot et trouver des méthodes pour décrire une image dans le but de retrouver ces objets dans un environnement hostile. Enfin, il m'a fallu gérer les informations sonores et auditives perçues par le robot de façon à ce qu'il construise sa base de connaissances commune avec l'utilisateur.

La première partie du rapport détaille le cadre de l'étude ainsi que la problématique. Puis, la deuxième partie décrit le travail de recherche ainsi que la conception du système. Ensuite, dans cette même partie, est expliquée la mise en application du système final et l'analyse des résultats de reconnaissance visuelle. Enfin, la dernière partie comprend les perspectives ainsi que des précisions sur la chronologie du stage.

1 Introduction au sujet de recherche

1.1 Présentation du cadre

1.1.1 L'INRIA

L'INRIA est un centre de recherche publique qui a pour objectif de créer un réseau de compétences dans le domaine de la recherche des sciences et technologies de l'information. Il est composé de huit centres répartis dans toute la France et compte environ 1000 chercheurs pour 1000 doctorants.

Les sujets de recherche sont divers :

- Mathématiques appliquées, calcul et simulation.
- Algorithmique, programmation, logiciels et architectures
- Réseaux, systèmes et services, calcul distribué
- Perception, cognition, interaction
- STIC pour les sciences de la vie et de l'environnement

L'équipe au sein de laquelle j'ai travaillé est liée à la partie perception, cognition et interaction.

1.1.2 L'équipe FLOWERS

L'équipe FLOWERS fait partie du Programme Actions Exploratoires. L'objectif de ce programme est de favoriser l'émergence de nouveaux sujets de recherche qui présentent un caractère exploratoire en rupture par rapport aux thèmes et approches traditionnelles.

Cette équipe travaille sur les sujets suivants :

- L'exploration et l'apprentissage intrinsèquement motivés : création de système basé sur la psychologie comportementale (curiosité, spontanéité, motivation, etc.) dans le but de réguler les comportements et de gérer les flux issus de l'apprentissage.
- Apprentissage social naturel et intuitif : développer des systèmes d'interaction et des mécanismes d'apprentissage pour permettre à un non-ingénieur d'enseigner naturellement à un robot.
- Découverte et abstraction de la structure d'ensembles non-interprétés de senseurs et de moteurs : étudier des mécanismes qui permettent à un robot de s'approprier un ensemble de connaissances sur ses canaux sensorimoteurs sans en connaître la sémantique.

Les robots apprennent des savoir-faire progressivement ; du plus simple au plus compliqué. Cette exploration peut être autoguidée par des comportements ou assistée par un acteur externe. Dans le cas où cet acteur est un humain, il faut concevoir des systèmes et des formalismes pour enseigner et guider le robot.

1.2 La robotique développementale

1.2.1 Définition générale et problématique

Afin de comprendre les problématiques et le travail effectué, il est important de définir ce qu'est la robotique développementale.

Elle a pour but de construire des systèmes aux développements autonomes. Pour cela, elle étudie des mécanismes qui permettent à des machines et à des robots d'apprendre des savoir-faire nouveaux pour pouvoir interagir et progresser dans des environnements physiques et sociaux initialement inconnus et changeants. Ce concept est à mettre en opposition avec les systèmes développés pour un contexte et un environnement particulier mais incapable de s'adapter au changement.

L'approche consiste à utiliser des mécanismes de la psychologie développementale dans des modèles robotiques. Il semble par exemple intéressant qu'un robot puisse apprendre de la même manière qu'un enfant : apprendre sur son corps, à travers les interactions avec l'environnement, etc. Ainsi, FLOWERS s'inscrit dans le domaine émergent de la robotique développementale ou « épigénétique ».

Les intérêts de cette démarche sont divers. Tout d'abord, elle permet de développer des systèmes beaucoup plus robustes aux variations de l'environnement. Ainsi, ce n'est plus le programmeur qui définit l'environnement de façon statique lors de la création du robot, mais le robot qui découvre son contexte de façon autonome. Par ailleurs, minimiser l'intervention de l'ingénieur une fois la conception initiale du système passée est aussi un objectif de ce travail.

Cet aspect est essentiel pour que les utilisateurs puissent partager des informations sur le monde avec un robot et renvoyer une impression de compréhension. Mais l'environnement, bien qu'il nous semble simple, devient très complexe dès que l'on tente de le formaliser pour que les robots puissent interagir avec lui. Or, apprendre des objets constitue la première étape de la découverte du monde pour les robots. Toutefois, les modes d'interactions actuels sont complexes et rigides, c'est cette situation qui rend difficilement accessible un certain nombre de technologies à beaucoup de personnes et d'applications.

1.2.2 Exemple

Pour que des utilisateurs non spécialistes puissent avoir des interactions avec un robot, il faut que les interfaces soient intuitives et très simple d'utilisation. Ce type d'interface peut s'avérer relativement complexe à créer. Par exemple, si l'on souhaite montrer des objets à un robot différents problèmes se posent : faire en sorte que le robot regarde l'objet, indiquer au robot l'objet voulu parmi tous les objets visibles, attirer l'attention du robot, imaginer un système qui fasse abstraction de la complexité technique, etc. La figure suivante illustre ces problèmes : comment un robot peut-il savoir ce qui est important dans ce qu'il voit et quel type d'information doit il utiliser.

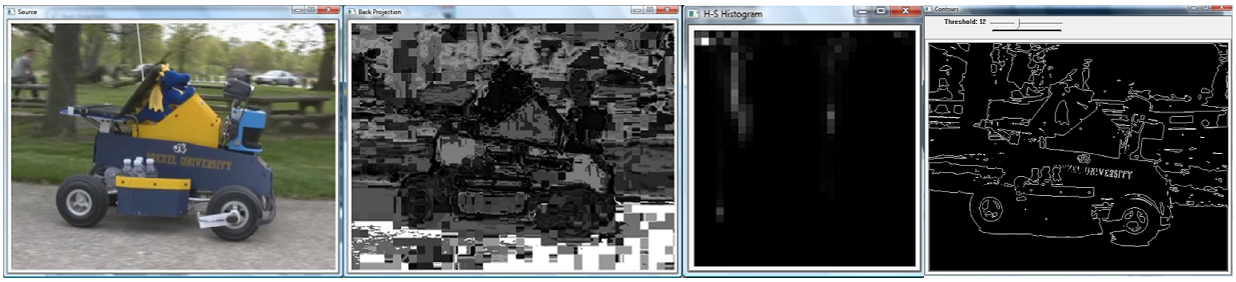


Figure 1 : Image visible et extraction d'information possible (*back propagation,* histogramme et contour)

Un doctorant de l'équipe, Pierre ROUANET, propose une solution pour répondre à certain de ces problèmes : utiliser une interface tactile d'IPhone. Ainsi, l'utilisateur voit en temps réel le flux vidéo issu de la caméra du robot. Puis, à l'aide de l'écran tactile, il fige l'image et entoure les objets sur l'écran. Avec une telle interface aucune connaissance ou formation n'est nécessaire pour échanger avec le robot. De plus, cela permet une interaction collaboratrice, c'est-à-dire de déléguer complètement le choix auprès de l'utilisateur : c'est l'utilisateur qui apporte ses propres connaissances au robot en lui indiquant ce qui est important dans l'image.

Figure 2 : Exemple IPhone

Un deuxième doctorant de l'équipe, Adrien BARANES, travaille sur la découverte intrinsèque motivée de « son corps » chez un robot. De cette manière, un robot peut découvrir son corps et comment l'utiliser. A terme, il pourra apprendre des savoir-faire et réagir en fonction de l'environnement. Ces méthodes sont à mettre en opposition avec celle dites statiques pour lesquelles un modèle est créé au départ et reste inchangé par la suite. Par exemple, un robot qui marche en appliquant une tension préenregistrée par un ingénieur mais qui serait incapable de s'adapter si des événements imprévus ont lieu, comme une jambe cassée, est un système statique.

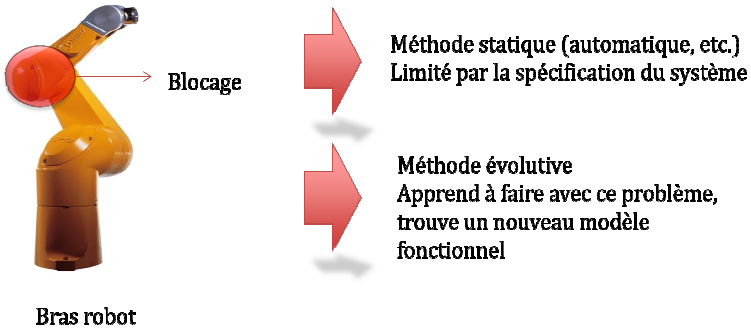


Figure 3 : Exemple d'évolutions possibles lors d'un blocage

La figure précédente illustre ce que peut apporter des modèles évolutifs et qui s'adapte au système en cas de problèmes ou d'évènements jusqu'alors inconnus.

1.3 Le sujet

1.3.1 Objectif

L'objectif de ce stage a été de travailler sur l'apprentissage de nouveaux objets à un robot en utilisant le son et l'image. Il s'agissait d'enseigner des objets pour ensuite les retrouver dans un environnement ou pour savoir quel objet est présent devant le robot. Les interactions avec le robot doivent être intuitives et possibles pour des personnes non spécialisées.

De plus, le robot doit apprendre en temps réel les informations apportées par l'utilisateur. Le système ne doit avoir besoin ni de pré-requis ni de catégories initiales. Cela signifie qu'avant de commencer l'apprentissage le robot ne connait aucun objet et construira les catégories auditives et visuelles au fur et à mesure.

1.3.2 Problématiques

Le sujet amène différents problèmes. Certains problèmes sont spécifiques à un domaine, tel que l'imagerie, tandis que d'autres sont liés à l'assemblage des technologies : quelle méthode utiliser pour traiter le son, faut-il le convertir en texte, peut-on associer uniquement un son à un objet, comment montrer l'objet voulu au robot et attirer son attention, quelle méthode utiliser pour décrire un objet visuellement, comment lier le son et l'image, comment créer des catégories, etc.

1.3.3 Travaux connexes

Ce stage étant un stage de recherche, un travail important a été fait au niveau de l'état de l'art. Le problème de l'apprentissage en robotique en utilisant l'association voix/image est un travail de recherche depuis déjà plusieurs années. Toutefois, ce travail dépend lourdement des avancés réalisées dans des domaines autres tels que l'imagerie ou encore les performances des systèmes. En effet, ce sont deux aspects qui peuvent faire progresser la vitesse et la qualité d'analyse et l'extraction des données. Mais les travaux diffèrent aussi de par leur finalité. Certains ont pour but de travailler sur les interactions, d'autres sur l'apprentissage et l'aspect linguistique et enfin des travaux ont pour objectif la création d'un système robuste.

En 2000, Luc Steels et Frederic Kaplan (Steels & Kaplan, 2000) enseignent à l'AIBO le nom de différents objets. Ils se focalisent sur l'aspect culturel et social. Par exemple, la construction de la signification et les méthodes d'apprentissage. Ils s'inspirent de l'apprentissage chez les enfants et tentent de reproduire les mêmes mécanismes avec un robot. Ils utilisent, comme nous le faisons, une interaction directe avec le robot. Mais, ils laissent l'aspect technique et reconnaissance de côté pour s'attarder davantage sur la psychologie. Notre travail diffère en ce point. En effet, nous nous attardons longuement sur la création d'un système de reconnaissance robuste en imagerie. De plus, puisque nous ne travaillons pas sur la signification, nous n'utilisons pas de reconnaissance vocale pour récupérer du texte. Enfin, notre système utilise les points d'intérêt (SURF) pour décrire les objets, alors que le système utilisé sur l'AIBO utilise un système d'histogrammes.

Par la suite, en 2002, Frank Lömker and Gerhard Sagerer (Frank & Gerhard, 2002) ont présenté leurs travaux sur un système multimodal qui associe des mots et la représentation des objets directement montrés par la main humaine. Leur système transcrit le son en texte et observe la gestuelle de la main. En fonction, de ces paramètres, il apprend des objets présents devant lui ou met à jour ses connaissances. Le système retrouve la main dans l'image en utilisant les propriétés et la reconnaissance de la peau humaine. Puis, les objets sont représentés à l'aide de différents histogrammes (couleurs, luminance, etc.) et structurés en graphe. Mais, à la différence de notre travail, ce système n'a pas pour objectif de rechercher et trouver des objets appris dans une scène quelconque. De plus, lors de l'apprentissage, les objets sont disposés sur un fond neutre. Toutefois, il est important de noter que ce système utilise une interaction tout à fait naturelle pour l'être humain qui montre des objets très souvent dans la vie de tous les jours.

Dans (Chen & Dana, 2004), un système est présenté pour tenter de répondre aux problèmes d'associations sons/images et de description visuelle d'un objet. Pour cela, ils utilisent un système de reconnaissance sonore robuste mais qui repose sur une base de données de plus 70 000 mots. Ce qui demande à avoir des pré-requis alors que nous tentons de travailler sans. De plus, la représentation visuelle des objets est faite grâce à des histogrammes de couleur. Enfin, lorsque le système apprend, plusieurs objets sont disposés devant lui. Il va alors associer les sons et les objets en fonction des fréquences des mots prononcés et des objets présents au moment ou les sons arrivent.

Une approche très utilisée pour apprendre la représentation visuelle des objets et créer des catégories est la méthode dite « sac de mots » (Gabriella, Christopher, Lixin, Jutta, & Cédric, 2004), (Kobus, Pinar, David, Nando, David, & Michael, 2003). Ce système est inspiré des systèmes utilisés pour la recherche de documents. Mais nous pensons que l'approche sac de mots présente un intérêt majeur lorsque l'on classifie les objets par catégorie. Par exemple, partons du principe que l'on possède 1000 images de voitures et 1000 images de maisons avec lesquelles on initialise la base du système. Lorsque que l'on veut tester une image pour savoir si elle représente une maison ou une voiture, le système de sac de mot est approprié. A ce moment là, l'approche sac de mot va permettre de faire ressortir des similitudes. Pour chaque catégorie, au sein d'une image, certaines parties seront très similaires à plusieurs images de la base. Mais cette méthode est inadaptée lorsque l'on cherche un objet particulier et que l'on possède déjà des images de cet objet dans la base. Après quelques tests, cette méthode a donc été abandonnée pour nous concentrer sur une comparaison image à image.

Toutefois, il est important de noter qu'à partir de là, ont émergé de nouveaux champs d'application. David FILLIAT (FILLIAT, 2007) propose un système utilisant l'approche sac de mots pour qu'un robot mobile puisse se déplacer et reconnaître les endroits où il se trouve. Ce robot peut alors se repérer et prendre des directions en fonction de ses objectifs.

D'autres recherches sont portées sur le langage et la signification des mots (Roy, 2005), (Kevin & Stephen, 2005). Dans ces recherches, les auteurs tentent de catégoriser les objets par leurs natures ou leurs fonctions. Le robot apprend des « objets » en interaction directe avec un humain. La représentation visuelle utilisée par le système est un histogramme de couleur. Mais dans ce travail, un modèle permet au système d'évaluer et de se représenter des concepts tels que « l'objet est à côté », « l'objet est dessus », etc.

En 2006, un système performant de classification/reconnaissance d'objets a été proposé (Heiko, et al., 2006). Mais ce système s'appuie sur une stéréovision et n'a pas pour objectif principal d'effectuer une association son/image. Il est d'abord question de créer un système de reconnaissance robuste. Cependant, ce système nécessite une caméra performante et des réglages précis ce qui le rendrait difficile à utiliser sur des robots ludiques.

D'autres travaux sont plus proches de nos recherches. Dans (OKA, NAKAFUSHIKI, & ITOH, 2005) est présenté un robot qui apprend la gestuelle que l'utilisateur lui fait subir en bougeant ses membres et l'associe avec un son obtenu en capturant un flux audio continu. Ce système associe mouvements et son avec une phase d'apprentissage et peut ensuite restituer ses connaissances.

Enfin, Iwahashi (Iwahashi, 2004) propose un système qui peut être perçu comme le plus proche du notre. Son système ne requiert pas de pré-requis : le système commence à apprendre sans rien connaître. De plus, l'apprentissage est complètement incrémental. L'utilisateur montre un objet devant une caméra stéréovision et prononce le nom de cet objet. Puis, la machine décide si elle connait le mot - si elle l'a déjà appris - ou non. Si elle ne connait pas le mot, elle ajoute un nouveau couple mot/image. En cas de « doute » elle peut demander confirmation à l'utilisateur ou corriger de faibles différences de son si l'image de l'objet semble être reconnue.

Mais notre système propose une approche différente. Tout d'abord nous ne convertissons pas le son en texte car bien que cela permette d'avoir des informations linguistiques supplémentaires sur un son, cela demande à posséder une base de phonèmes et une phase d'apprentissage.

Enfin, notre système utilise une caméra mono-vision et doit pouvoir retrouver des objets dans une scène quelconque. De plus, nous utilisons des descripteurs issus de recherches récentes. Ce qui nous a amené à créer notre propre système de traitement et de reconnaissance d'image.

2 Méthodologie

2.1 Traitements sonore

Dans cette partie, nous expliquons les aspects du travail liés directement au traitement du son : extraction/traitement/comparaison. Nous utilisons l'analyse PLP qui est une méthode très utilisée dans le domaine de la reconnaissance vocale.

2.1.1 Choix et représentation

Dans notre système, nous ne convertissons pas le son en texte. En effet, notre objectif est que le robot commence sans le moindre pré-requis. Or, les systèmes actuels de reconnaissance vocale nécessitent une base de phonèmes et bien souvent une période d'apprentissage pour chaque interlocuteur.

En contre partie, nous ne tirons aucune information autre que le son. Il serait par exemple difficile de travailler sur la signification ou l'aspect linguistique sans une conversion en texte. Le son entendu n'est alors pas obligatoirement une voix, cela peut être un bruit ou toute autre chose.

2.1.2 Capture du son

Le flux sonore est capturé en continu depuis un microphone. Ce module a été développé avec le logiciel Matlab car de nombreuses fonctions mathématiques nécessaires pour le traitement y sont déjà présentes. Il va découper le flux en fonction du silence. Ainsi, lorsque l'interlocuteur parle puis s'arrête, le système va considérer l'ensemble comme un « mot ». Puis, lorsque l'interlocuteur recommence à parler alors c'est un nouveau son qui commence.

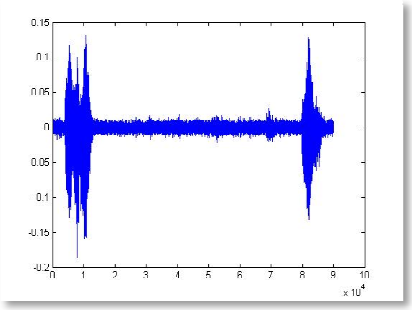


Figure 4 : onde sonore « livre noir [silence] livre »

2.1.3 Traitements

Pour pouvoir travailler avec les sons capturés et effectuer des comparaisons, nous utilisons l'Analyse PLP (*Perceptually based Linear Prediction analysis*). La prédiction linéaire perceptuelle exploite les connaissances du système auditif humain pour paramétrer la parole en introduisant des mécanismes psycho acoustiques de l'oreille humaine. Dans cette analyse, les coefficients sont calculés suite à la résolution d'équations obtenues par une succession de traitements présentés en annexe. L'analyse RASTA (*RelAtive SpecTrAl*) a pour but de supprimer les variations temporelles

trop lentes ou trop rapides correspondantes au bruit. Elle se base sur le fait que la perception humaine réagie aux valeurs relatives plus qu'aux valeurs absolues. Cela rend le système d'avantage robuste aux perturbations (Daniel & W., 2005) (H. & N., 1994) (Hermansky, 1990). Grâce à cette décomposition, nous obtenons une représentation du son en fenêtres spectrales.

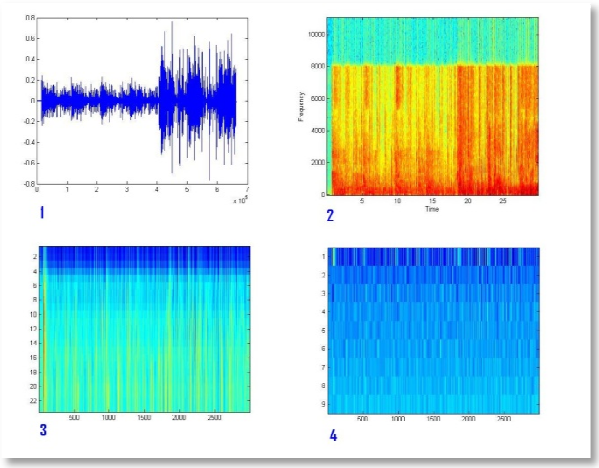


Figure 5 : Exemple PLP-Rasta

Légende de la figure :

1 – « Hold Me Now »

2 – Spectrogramme

3 – Spectrogramme PLP-Rasta

4 – Cepstra PLP-Rasta

Enfin, après avoir appliqué PLP-Rasta, nous comparons deux sons en utilisant DTW (*Dynamic Time Warping*) (L. R. & B., 1993). Cet algorithme donne la distance entre deux sons. Dans notre système, nous considérons que si cette distance est inférieure à un seuil alors deux sons sont les mêmes.

2.2 Traitement de l'image

Le traitement de l'image est réalisé en C++ et repose sur la librairie OpenCv. Cette librairie créée à l'initiative de INTEL est une librairie « *open source* » multiplateformes de traitement d'images et d'accès aux périphériques vidéo.

Comme nous l'avons expliqué de nombreux travaux ont été faits dans le domaine de la reconnaissance d'objets. Mais, de par les objectifs que nous nous sommes fixés - apprentissage sans pré-requis, apprentissage instantané, restitution rapide, recherche dans un environnement hostile – nous avons été amenés à créer nos méthodes propres pour répondre aux différents problèmes rencontrés.

2.2.1 SURF/SIFT

SIFT (*Scale-invariant feature transform*) est un algorithme qui permet de détecter et de décrire des descripteurs locaux dans une image. C'est-à-dire qu'à partir d'une image, l'algorithme va trouver des points d'intérêt qu'il va décrire à l'aide de différentes propriétés. Cet algorithme a été proposé par David LOWE (Lowe D. G., 1999). Il est actuellement breveté et sous la propriété de l'université de Columbia. Cet algorithme repose sur des formules mathématiques complexes qu'il n'a pas été utile d'étudier dans ce stage.

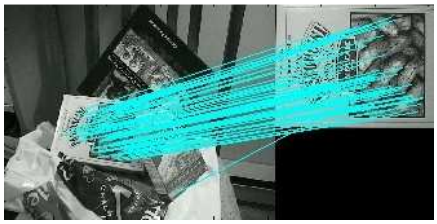


Figure 6 : Exemple de correspondance SIFT

SURF (*Speeded Up Robust Features*) est un descripteur basé en partie sur SIFT proposé en 2006 (Herbert, Tinne, & Luc, SURF: Speeded Up Robust Features, 2006). SURF est beaucoup plus rapide que SIFT. Néanmoins, d'après nos différents essais, SURF trouve moins de descripteur que SIFT pour des images identiques. Durant mon stage, j'ai utilisé SURF car le traitement est nettement plus rapide. De plus, des librairies SURF libres et gratuites sont disponibles.

Toutefois, ces méthodes traitent entièrement une image. Il faut donc indiquer au système quels sont les points importants et ceux qui ne le sont pas, comme par exemple ceux issus de l'arrière plan.

Chaque descripteur SURF possède les propriétés suivantes : position dans l'image, orientation, un vecteur descripteur de 64 valeurs et un niveau d'échelle. Dans la partie 2.2.3 traitant de la correspondance sont explicitées les propriétés importantes que nous utilisons.

2.2.2 Extraction de contour

Dans un premier temps, notre système requiert une phase d'apprentissage. Pour que le robot puisse enregistrer les images d'un objet, nous avons créé un système qui permet d'extraire les formes en mouvement. Pour cela, le moteur de capture attend que l'image soit stable. Puis, il indique à l'utilisateur qu'il est prêt à enregistrer les images. L'utilisateur agite et montre l'objet devant la caméra. Grâce à la différence entre l'arrière plan et l'utilisateur, le système sait où il doit porter de l'intérêt. Si le robot bouge, si il est secoué ou si une trop grande zone filmée est en mouvement, le robot indique qu'il n'apprend plus jusqu'à ce que l'image soit stable de nouveau.

L'algorithme qui permet de faire la différence entre deux images (le fond et l'image contenant l'objet) et qui trouve le contour prend 3 ms sur un ordinateur équipé d'un processeur double cœur. Le système capture, pour chaque objet, une liste d'images et de points SURF. Il permet d'obtenir une forme convexe qui sera la zone d'intérêt.

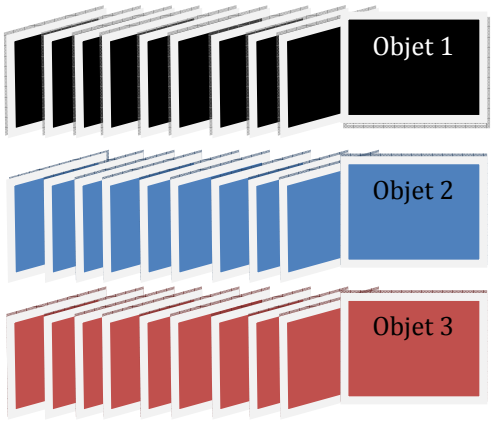
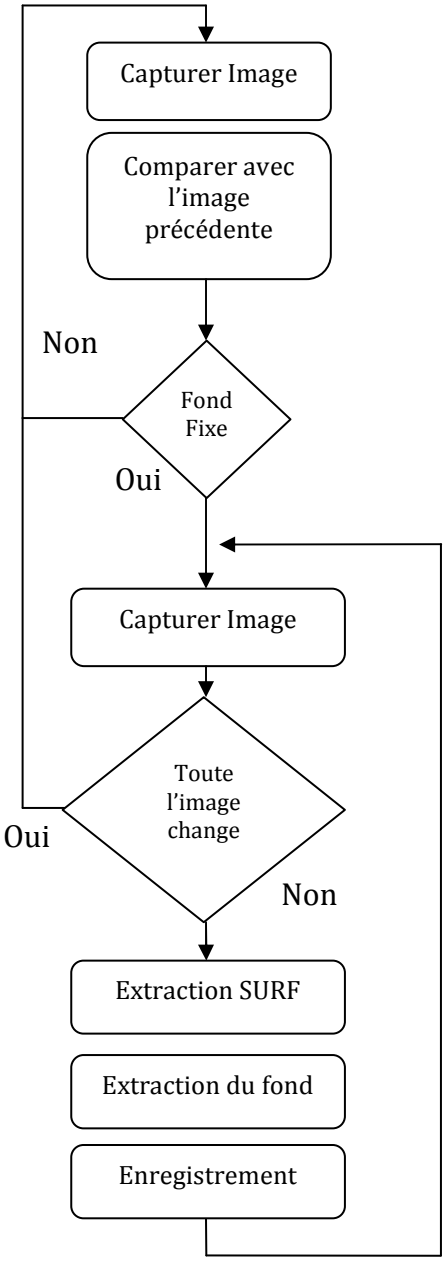
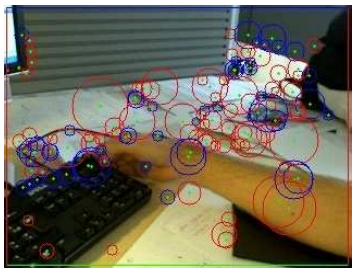


Figure 7 : Représentation de la base de données visuelle

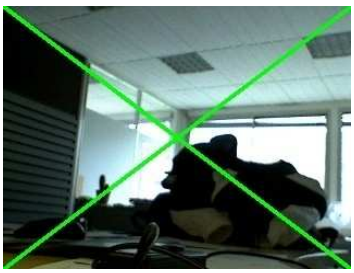
Ainsi, pour chaque objet appris, le système possède un film. Pour chaque image de ces films, il sait extraire les descripteurs SURF et filtrer ceux appartenant à l'objet des autres. Ainsi, un objet est représenté en mémoire par une liste de groupes de descripteurs SURF.

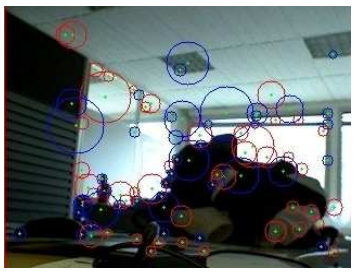
La figure suivante montre les différents états du système d'extraction et de contour. Lorsqu'une croix rouge est présente sur l'image cela signifie que le système n'est pas prêt à apprendre. Puis, une fois que le fond est stable, le système indique qu'il est prêt par une croix verte. Enfin, lorsqu'on met un objet devant la caméra, le système fait le contour et ne garde que les descripteurs SURF à l'intérieur. Les points SURF sont représentés par des pixels verts et entouré par des cercles rouges ou bleus proportionnels à leurs intensités.



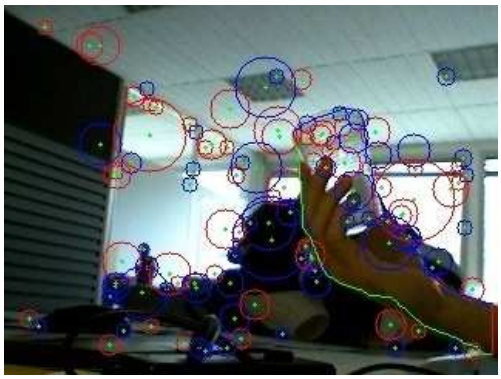


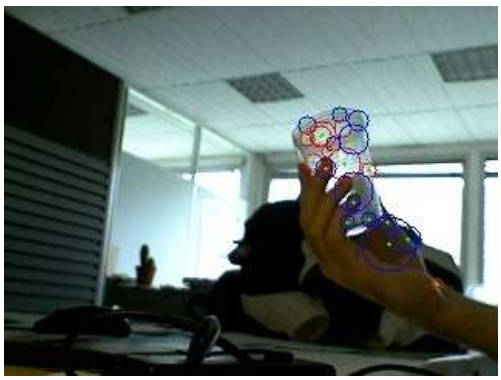
1 – Image entière en mouvement, le robot n'enregistre pas





2 – Fond fixe, l'utilisateur peut montrer les objets voulus





1 – L'utilisateur montre un objet, le contour permet alors d'extraire les bons points SURF

Figure 8 : Principe d'extraction du contour

2.2.3 Correspondances

Une fois que les objets sont enseignés, il faut pouvoir tester la ressemblance entre une image test et toute la base de connaissances. Or, les seules informations que le système connait sur les images sont les descripteurs SURF. Une distance peut être calculée entre deux descripteurs SURF en utilisant la norme de la différence entre les deux vecteurs des descripteurs :

$$Dist(P1, P2) = || P1 - P2 || = \sqrt{\sum_{i=0}^{64} (P1[i] - P2[i])^2}$$

Ainsi, plus la distance est proche de 0 plus les descripteurs SURF sont issus de zones ressemblantes. Toutefois, cette mesure ne permet pas de savoir directement si deux images comparées sont les même ou non. En effet, cela permet de savoir si, dans deux images différentes, deux zones sont plus ou moins identiques. Mais cela n'indique pas si c'est le même objet qui est présent sur ces images. D'ailleurs, selon les photos, des descripteurs issus d'images représentant des objets différents peuvent être plus proches que des descripteurs représentant un même objet.

La figure suivante illustre la difficulté de correspondance entre descripteurs. Ainsi, en comparant un descripteur SURF issu d'un carré avec trois autres images, dont deux représentent également un carré, on peut voir que la distance peut différer même si le point comparé nous semble être le même. De plus, la distance peut être très faible entre deux descripteurs représentant une zone similaire mais issus de deux objets différents. Cela pose un problème car cela signifie qu'on ne peut pas se contenter de relier les descripteurs SURF proches entre eux.

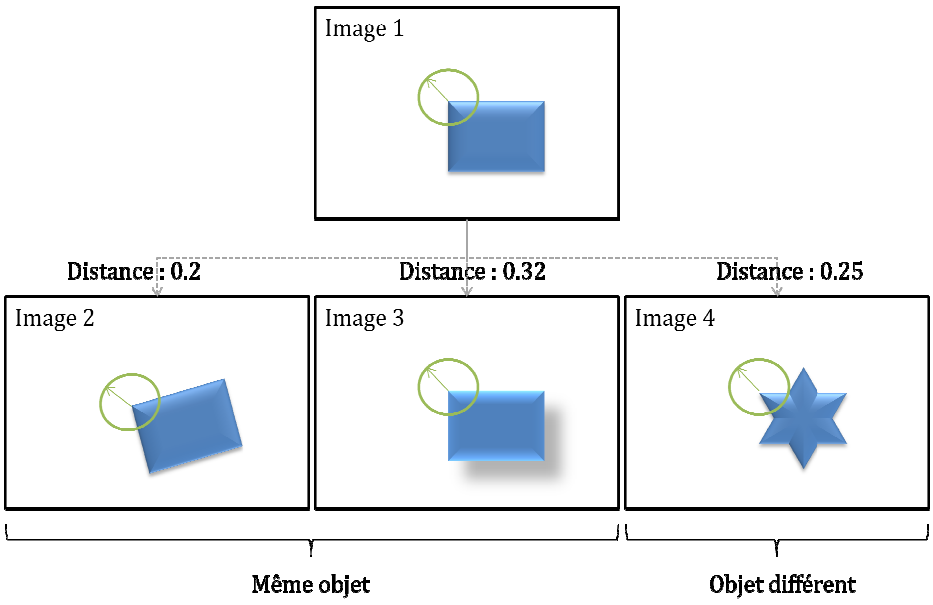


Figure 9 : Problème de correspondance

Différents algorithmes permettent de mettre en correspondance des groupes de points SURF. Nous présentons ici deux algorithmes très populaires ainsi que celui que nous avons créé pour répondre en partie à ce problème.

- **Méthode proposée par David LOWE**

Comme cela a été indiqué précédemment, l'algorithme SURF est basé en partie sur l'algorithme SIFT. Une méthode de correspondance répandue est celle proposée par David LOWE dans son article présentant SIFT pour la première fois (Lowe D. G., 1999).

Cette méthode parcourt tous les descripteurs de la première image et les compare aux descripteurs de la seconde image en gardant en mémoire les deux distances les plus petites. Puis, on utilise le ratio entre ces deux plus petites distances pour savoir si le descripteur de l'image 1 en cours doit être relié à celui de l'image 2.

```
Pour tous les points P1 de l'image 1
| min_dist  = MAX
| min_dist2 = MAX
| index     = -1
|
| Pour tous les points P2 de l'image 2
| | dist_p1_p2 = dist(P1,P2)
| |
| | si dist_p1_p2 < min_dist
| | | min_dist = dist_p1_p2
| | | index    = p2
| | sinon si dist_p1_p2 < min_dist2
| | | min_dist2 = dist_p1_p2
| | fsi
| fpour
|
| si min_dist < min_dist2 x 0.5
| | ajouter_correspondance(p1 , index);
| fsi
|
fpour
```

Algorithme 1 : Méthode de comparaison proposé par David LOWE

Toutefois, cette méthode présente plusieurs problèmes :

- Un descripteur de l'image 2 peut être relié à plusieurs descripteurs de l'image 1. Bien que cela ne soit pas gênant dans toutes les situations, cela pose un problème conceptuel et lorsque nous utilisons le filtre spatial.
- Si un descripteur de l'image 1 ressemble à plusieurs descripteurs de l'image 2 aucun lien ne sera fait. Or dans un même objet, il se peut que plusieurs zones soit très proches.
- Le résultat n'est pas le même si on compare l'image 1 à l'image 2 ou l'inverse.

La figure suivante présente des résultats utilisant la méthode de correspondance décrite par D. LOWE. Sur chaque image, on peut voir les points SURF représentés par des pixels verts et entourés par des cercles rouges ou bleus décrivant l'intensité des descripteurs. Les lignes vertes sont le résultat de la mise en correspondance.

Figure 10 : Exemples d'illustrations de la méthode de David LOWE

- **Méthode utilisant un seuil**

Cette méthode est la première qui vient à l'esprit : deux points correspondent si la distance est inférieure à un seuil. Toutefois, pour chaque point de la première image uniquement la meilleure correspondance est retenue.

```
Pour tous les points P1 de l'image 1
| index    = -1
| min_dist = MAX
|
| Pour tous les points P2 de l'image 2
| | dist_p1_p2 = dist(P1,P2)
| | si dist_p1_p2 < min_dist
| | | min_dist = dist_p1_p2
| | fsi
| fpour
|
| si min_dist < SEUIL
| | matches.ajouter_correspondance();
| fsi
|
fpour
```

Algorithme 2 : Méthode de correspondance par SEUIL

Cette méthode présente elle aussi plusieurs problèmes. Plusieurs descripteurs de l'image 1 peuvent correspondre à un descripteur de l'image 2 et le résultat n'est pas le même si on compare l'image 1 avec l'image 2 ou l'inverse. De plus, il est très difficile de trouver un seuil fonctionnant dans la majorité des cas.

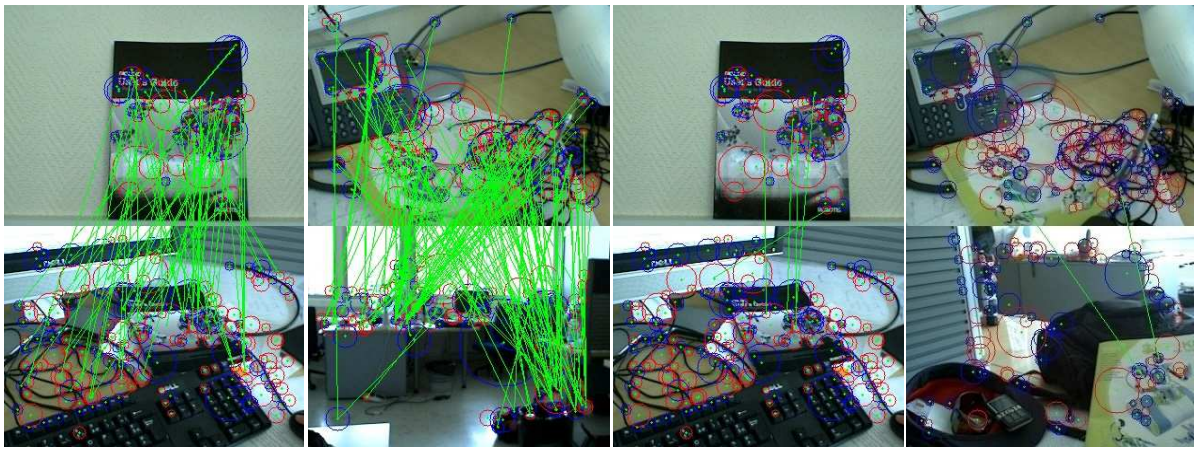


Figure 11 : Exemples d'illustrations par la méthode de SEUIL (0.6 et 0.3)

- **Méthode mélangeant bonnes correspondances et seuil**

Nous nous inspirons des problèmes rencontrés avec les méthodes présentées précédemment pour créer notre propre algorithme de correspondance.

Nous calculons toutes les distances entre les points P1 de l'image 1 et les points P2 de l'image 2. Une correspondance entre P1 et P2 est faite si trois conditions sont respectées. Il faut que la distance entre P1 et P2 soit inférieure à un seuil. Il ne faut pas que l'un des deux descripteurs soit déjà relié avec une distance inférieure à la distance P1-P2. Enfin, il ne faut pas qu'un meilleur descripteur soit libre aussi bien pour P1 que pour P2.

```
Pour tous les points P1 de l'image 1
|
| Pour tous les points P2 de l'image 2
| | dist_p1_p2 = dist(P1,P2)
| | si dist_p1_p2 < SEUIL
| | | correspondance[P1].ajouter_correspondance_triees(P2)
| | fsi
| |
| fpour
|
Fpour

Changement = VRAI
Meilleures_correspondances[ points1.taille ]

Tant que changement faire
|
| Pour tous les points P1 de l'image 1
| | |
| | si !correspondance[P1].est_vide() ET
    Meilleurs_correspondances[correspondance[P1].tete] != P1
| | | |
| | | Tant que !correspondance[P1].est_vide() ET
      (Meilleurs_correspondances[correspondance[P1].tete] OU
       Meilleurs_correspondances[correspondance[P1].tete].dist
                                                  < correspondance[P1].tete.dist)
| | | |  correspondance[P1].pop
| | | ftq
| | | |
| | | si !correspondance[P1].est_vide()
| | | | si !correspondance[P1].est_vide()
```

```
| | | | | Dépiler matches[P1] jusqu'à trouver une correspondance
| | | | fsi
| | | fsi
| fpour
|
ftq
```

Algorithme 3 : Méthode de la meilleure correspondance possible

Cette méthode fournit aussi des correspondances erronées et demande plus de calculs. Néanmoins, elle permet d'obtenir un nombre important de correspondances uniques et robustes.

Figure 12 : Exemple d'illustration par la méthode de la meilleure correspondance possible

Comme il a été expliqué, certaines correspondances sont « fausses ». C'est-à-dire qu'en testant la présence d'un objet dans une photo, deux descripteurs vont correspondre alors qu'ils ne viennent pas du même objet ou du même endroit. D'un point de vue mathématique, ces correspondances sont tout aussi justes que les autres. En revanche, d'un point de vue de la signification elles sont erronées. Il faut alors tenter de les supprimer, c'est le rôle des filtres.

2.2.4 Filtre spatial

Le principe du filtre spatial a été proposé par SIVIC et ZISSERMAN dans (Josef & Andrew, 2008). L'idée est d'observer dans chaque image les voisins de tous les descripteurs. Puis, une fois que les correspondances ont été faites en utilisant l'algorithme présenté précédemment, on regarde si les voisins correspondent aussi. Cet algorithme a deux paramètres : le nombre de voisin à prendre en compte et le nombre de voisins qui doivent correspondre pour considérer que la correspondance est bonne.

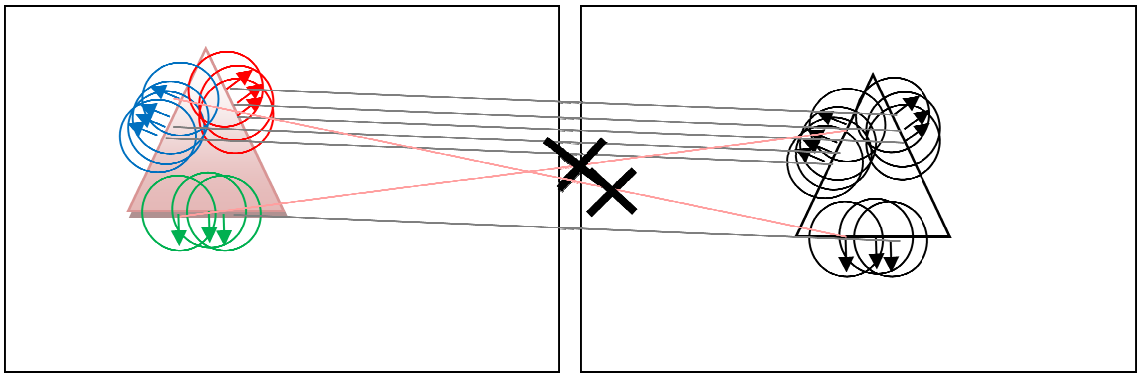


Figure 13 : Principe du filtre spatial

Cette méthode permet d'éliminer une partie des erreurs de correspondances. Mais elle nécessite un nombre important de points (plus de dix) et il ne peut y avoir qu'une correspondance par descripteur sinon la notion de voisin est faussée.

```
# On trouve les voisins des points de l'image 1 qui ont une correspondance
# avec des points de l'image 2
Pour p1 dans points_img1
| si exist_correspondance( p1 )
| | voisins1[ p1 ] = trouver_N_voisins( points_img1 )
| fsi
fpour

# On fait de même avec l'image 2
Pour p2 dans points_img2
| si exist_correspondance( p2 )
| | voisins2[ p2 ] = trouver_N_voisins( points_img2 )
| fsi
Fpour

# Puis on fait un choix en fonction du rapport entre les voisins trouvés dans
# chaque image
Pour p1 dans points_img1
| si exist_correspondance( p1 )
| ET nb_voisins_correspondants( voisins1[ p1 ] , voisins2[ correspondant(p1) ] ) < V
| | supprimer_correspondance( p1 )
| fsi
fpour
```

Algorithme 4 : Filtre spatial

2.2.5 Filtre angulaire

Le filtre angulaire a le même objectif, c'est-à-dire enlever des correspondances qui sont erronées. Pour cela, on calcule la différence d'angle entre les descripteurs qui correspondent puis on la normalise dans l'intervalle [0;2π]. On trouve ensuite l'orientation majoritaire et on ne garde alors que les correspondances dont le changement d'orientation est proche de cette orientation.

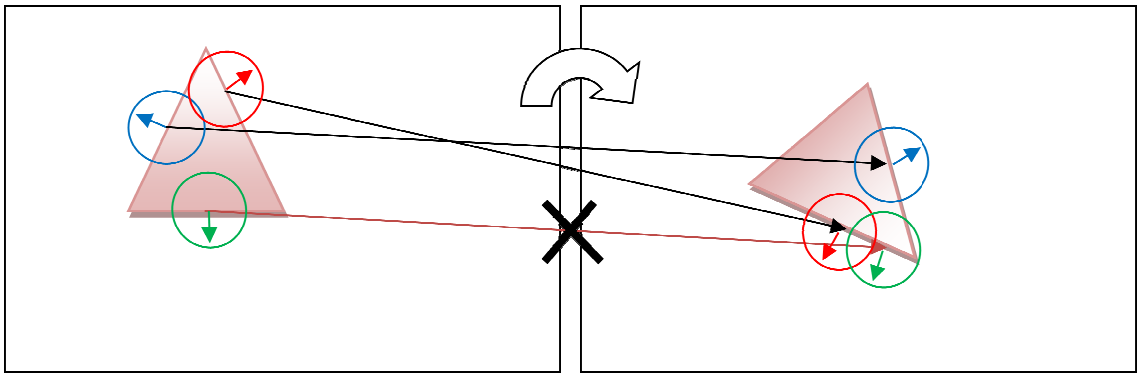


Figure 14 : Principe du filtre angulaire

2.2.6 Critère de choix

Nous avons montré comment créer les correspondances entre deux images et comment les filtrer. Toutefois, il reste à décider quand est-ce que l'on doit considérer qu'un objet présent devant la caméra est connu dans la base de connaissances.

- **Prise en compte de tout le film**

Dans un premier temps, nous avons testé des approches globales. C'est-à-dire que nous comparions une image de test avec toutes les images des films des objets présents dans la base de connaissances. Et nous décidions quel objet était devant la caméra en prenant en compte le résultat global obtenu entre l'image test et un film.

Par exemple, l'approche la plus simple consistait à sommer le nombre de correspondances trouvées dans un film après comparaison entre l'image test et toutes les images du même film. Le film qui présentait le plus de correspondances était choisi. Différentes approches similaires ont été testées : pondérations, filtre temporel, etc. mais aucune n'a donnée de résultat vraiment significatif.

Une deuxième approche basée sur TFIDF (*term frequency-inverse document frequency*) et sur la méthode sac de mots a été adaptée à notre problème. La méthode « sac de mots » est inspirée des moteurs de recherche sur internet lorsque l'on cherche à comparer des documents ou à trouver un document lié à des mots-clés. Toutefois, cette méthode est de plus en plus utilisée dans le traitement et la classification d'images. Or nous avons tenté différentes approches et nous présentons ici une approche « sac de mots » modifiée car au lieu de créer un vocabulaire en fonction des informations contenues dans tous les films, nous créons le vocabulaire à partir de notre image de test.

Dans notre cas, chaque film est représenté par un vecteur dont la taille est égale au nombre de descripteurs SURF dans l'image de test :

$$Vf = (t1, \dots, tD) \quad D: nombre\ de\ descripteurs$$

Ce vecteur représente le vocabulaire d'un film. Les mots sont les descripteurs SURF trouvés dans l'image de la caméra c'est-à-dire l'image de test. Nous regardons combien de fois ces descripteurs sont aussi présents dans les documents c'est-à-dire les vidéos. Pour cela, il faut comparer l'image de

test avec toutes les images de base de connaissance avec les méthodes expliquées précédemment. Puis, nous remplissons le vecteur de chaque film à l'aide de la formule suivante :

$$td = \frac{nid}{nd} log\frac{N}{Ni}$$

nid : le nombre d'occurrences du descripteur SURF i dans le film (le nombre de fois que ce descripteur a correspondu dans une image du film)

nd : le nombre de descripteurs trouvés dans le document (combien de descripteurs de l'image de test ont correspondu au moins une fois)

N : le nombre de films dans la base (le nombre d'objets appris)

Ni : le nombre de films contenant le descripteur SURF i (le nombre de films qui ont trouvé au moins une correspondance avec ce descripteur SURF)

L'image de référence contient tous les mots une fois, puisqu'elle contient tous les descripteurs, il a donc pour vecteur :

$$Vref = (1, \dots, 1)$$

La distance entre l'image et un film est alors calculée par :

$$sim(Vf, Vref) = Vf \intercal Vref$$

Mais ces approches n'ont pas donné de résultats suffisants hormis pour des objets particuliers. Nous avons alors tenté une approche différente.

- **Comparaison image à image**

Par le suite, nous sommes partis du postulat suivant : « si un objet est présent dans un film de notre base, il y a une image et une seule pour laquelle la comparaison apporte un très grand nombre de correspondances ». L'idée est de trouver dans la base de films la meilleure image et de considérer que c'est l'objet que l'on connait.

```
# on calcule les points surf pour l'image de test
surf_in = calculer_surf( image_test )

max_corresp = 0
max_f = -1

# Pour chaque film de la base et pour chaque image
Pour f de 0 à films.taille
| Pour i de 0 à f.nbImages
| | # on calcule le nombre de correspondance avec l'image test
| | nb_corresp = compter( correspondance( surf_in , film[f].image[i] ))
| | si nb_corresp > max_corresp
| | | max_corresp = nb_corresp
```

```
| | | max_f = f
| | fsi
| fpour
fpour

# si le nombre de correspondances est supérieur à un seuil on considère
# que l'objet est dans la base
si nb_corresp > SEUIL
| afficher : l'objet correspondant est 'max_f' avec 'max_corresp' correspondances
Sinon
| afficher : objet non reconnu
fsi
```

Algorithme 5 : Meilleure image

Ainsi, on retrouve l'image de la base la plus proche de l'image de test. Puis, si le nombre de correspondances entre ces deux images est supérieur à un seuil, on indique que l'on connait l'objet, sinon on indique que l'objet n'est pas reconnu. Ce seuil est donc important, il va être un critère de décision. En effet, si sa valeur est élevée il sera plus difficile pour une image d'être reconnue. Mais conduira à un système plus robuste. En revanche, si le seuil est faible, il y aura davantage d'erreurs mais il sera facile de retrouver un objet. Des tests ont été faits sur ce seuil et sont présentés dans la partie suivante.

2.3 Résultats de la reconnaissance visuelle

Pour tester le système nous lui avons enseigné 9 différents objets. Puis, nous avons pris 21 photos de chacun de ces objets à différents endroits et nous avons demandé au système ce que représentent ces photos.

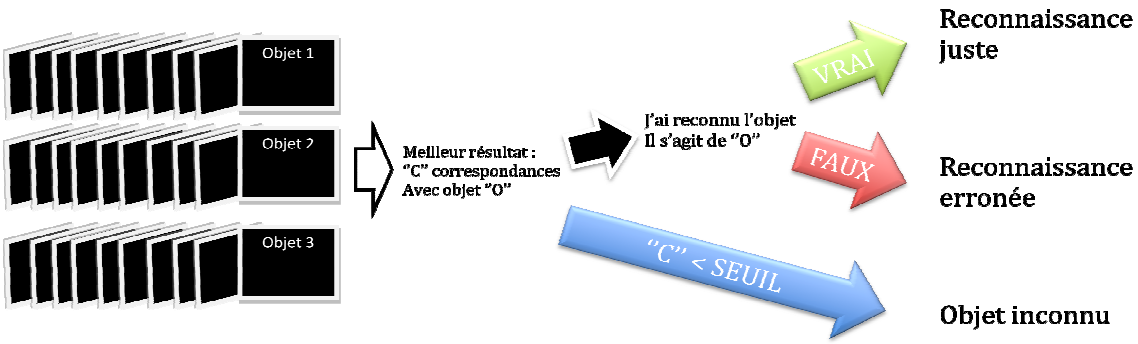


Figure 15 : Principe de reconnaissance

Pour chaque image testée il y a trois résultats possibles : vrai, faux ou inconnu. Le seuil est donc très important car il permet de filtrer les mauvais résultats.

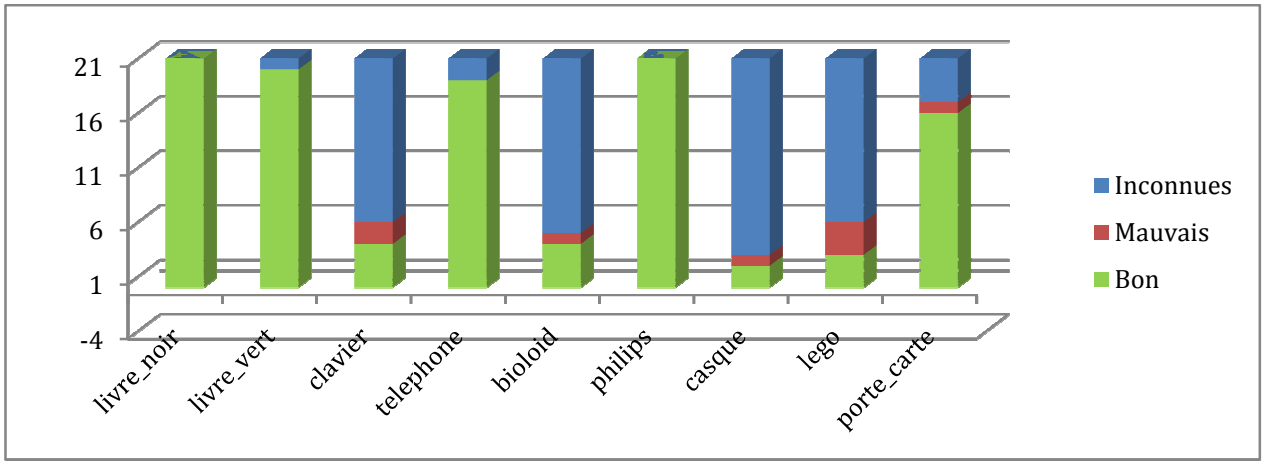


Figure 16 : Résultats du test de reconnaissance

Les résultats montrent que certains objets sont très bien reconnus tandis que d'autres ne le sont jamais. Les cinq objets qui sont reconnus (livre vert, livre noir, téléphone, carton Philips et porte carte) sont des objets qui présentent énormément de descripteurs SURF et qui sont relativement plats. A l'inverse, le Bioloid, le casque et le lego ont beaucoup moins de descripteurs SURF.

Il y a un nombre important de situations dans lesquelles des objets ont des correspondances avec l'arrière plan des photos de ces 3 objets plutôt que les objets eux-mêmes. Pour ce qui est du clavier, cet objet est constitué de multiples touches toutes semblables. Lorsque l'on teste deux images contenant un clavier il y a de nombreuses correspondances. Mais le filtre spatial va en supprimer une très grande partie puisque qu'une touche peut correspondre avec n'importe quelle autre.

Figure 17 : Illustration des résultats de comparaisons

Parallèlement, des tests ont été faits pour définir un seuil de choix optimal. La figure suivante présente l'évolution des résultats en fonction de ce seuil. Le seuil de 20 correspondances permet de diminuer de beaucoup la marge d'erreur tout en conservant de nombreux bons résultats. En revanche, en prenant un seuil de 24 il n'y a plus d'erreur mais dans le même temps cela va supprimer un certain nombre de résultats qui étaient correctes.

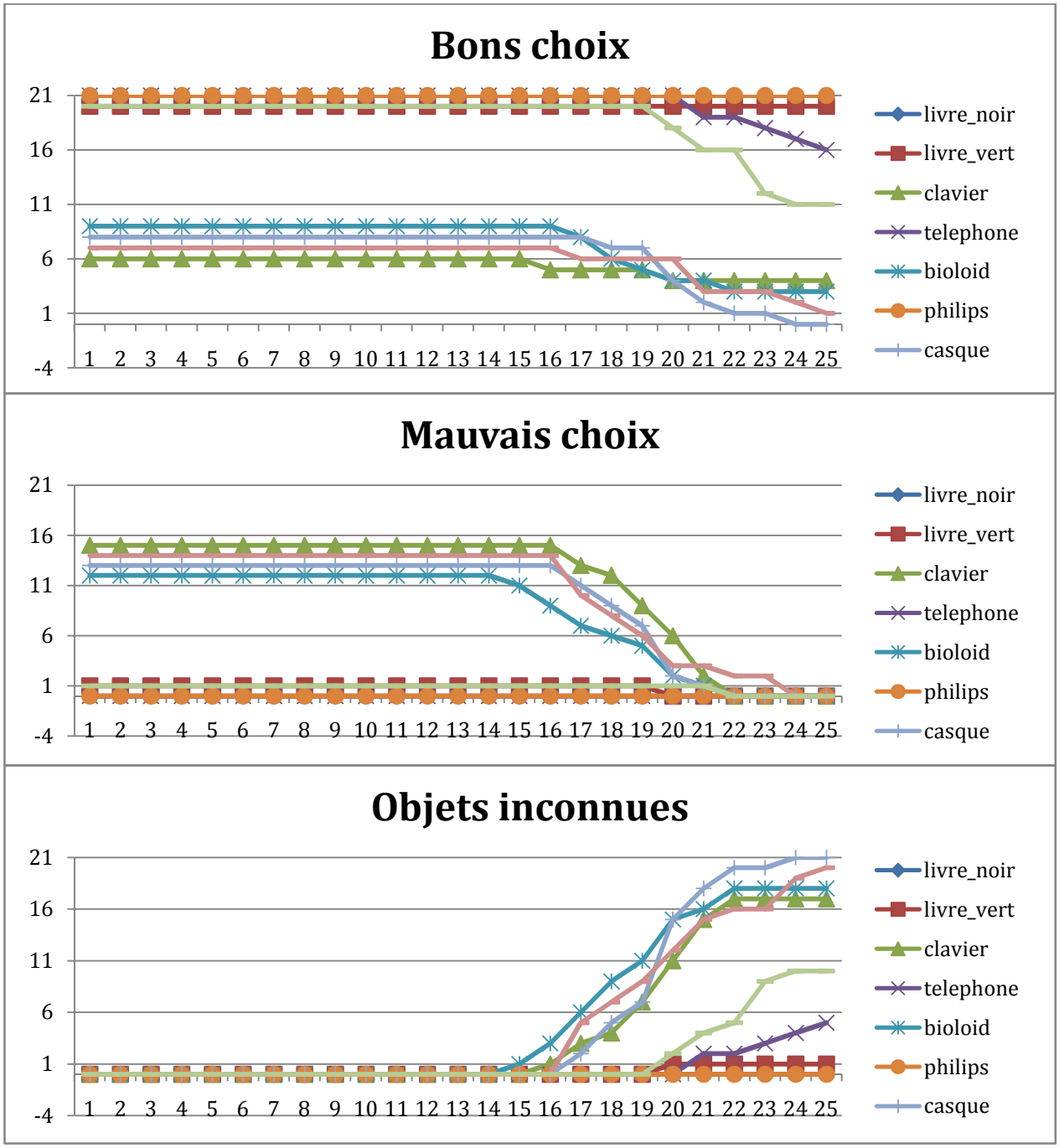


Figure 18 : Variations des résultats en fonction du seuil de correspondances

Sur la figure précédente, on peut observer les résultats en fonction du seuil. Par exemple, le carton Philips a toujours un nombre de correspondances supérieur au seuil, il est donc toujours reconnu correctement. En revanche, les photos de test du Bioloid correspondent soit avec un autre objet et donnent de mauvais résultats soit elles ont un nombre de correspondances proche de 18. Lorsque le seuil est à cette valeur, le système indique ne pas reconnaître des images du Bioloid alors qu'elles étaient correctement reconnues pour un seuil inférieur.

2.4 Phase d'apprentissage et association

2.4.1 Déroulement de l'apprentissage

Comme il a été expliqué dans la partie précédente, notre système possède deux modules d'enregistrement/comparaison, un pour le son et l'autre pour l'image. Lors de l'apprentissage, ces deux modules sont utilisés simultanément. Nous avons cherché à formaliser l'apprentissage de façon à organiser le lien entre ces deux modules. Cela permet de répondre à différents problèmes : quand enregistrer le son, quand est ce qu'un objet est montré, comment lier le son et l'image, etc.

Lors de l'apprentissage, le système peut recevoir en entrée des sons et des images. Nous avons considéré que pour enseigner un « objet » il fallait prononcer des mots et montrer l'objet en même temps au système. Ce comportement est très similaire à celui qu'un être humain peut avoir avec un nouveau né en voulant lui apprendre quelque chose. Mais dans certain cas, notre système peut entendre des sons sans que rien ne soit présent devant la caméra ou l'inverse. Dans de telles situations, nous avons décidé que le système n'apprenait pas.

De cette manière, comme cela est explicité sur la figure suivante, si on parle au robot sans lui montrer d'objet, le système ne prendra pas en compte les sons entendus. De même, si on agite un objet sans parler et que l'on retire l'objet de devant la caméra, les informations ne seront pas gardées par le système car aucun son n'aura été perçu pendant que l'objet était filmé. En revanche, si on commence à montrer un objet devant la caméra et que l'on prononce quelques mots, puis que l'on retire l'objet, alors ces données sont apprises par le système.

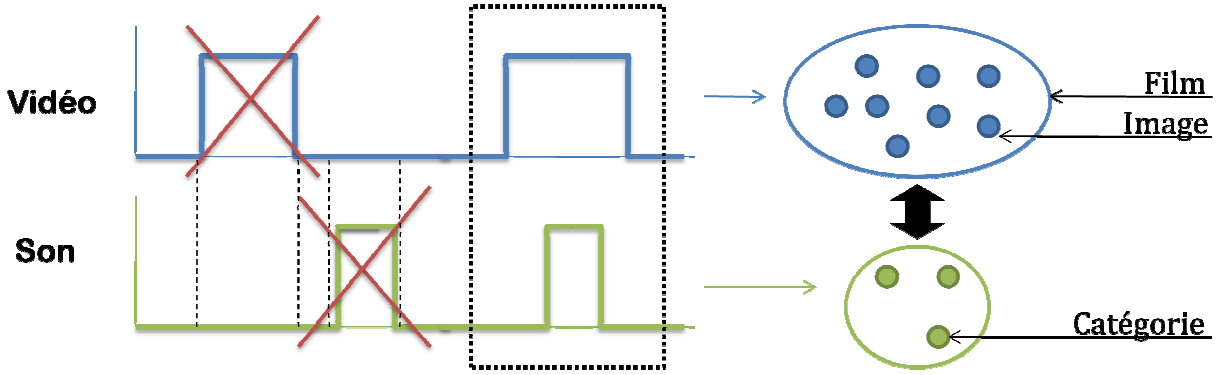


Figure 19 : Phase d'apprentissage

Ce formalisme ajoute de la robustesse. Ainsi, si quelqu'un passe devant la caméra mais qu'il n'y pas de bruit rien ne sera gardé. Enfin, pour définir un seul objet, plusieurs phases d'apprentissage peuvent avoir lieu. Dans ce cas, les informations seront concaténées.

2.4.2 Relation sons - images

Durant la phase d'apprentissage, plusieurs sons peuvent être prononcés pour le même objet. Grâce à ces sons, le système va générer automatiquement des catégories (une catégorie par son). On peut alors créer une relation entre l'espace visuel et l'espace acoustique. Ainsi, un objet est composé par

plusieurs images uniques et propres à lui-même mais aussi plusieurs sons qu'il peut partager avec d'autres objets.

Par exemple, si on enseigne au robot deux livres. Supposons que pour le premier, nous prononcions deux sons en le montrant: « livre … livre vert ». Pour le deuxième, nous prononcions 3 sons : « livre bleu … livre … grand ». Alors, ces deux objets vont partager une catégorie dans l'espace acoustique : « livre » car il a été prononcé dans les deux cas.

A partir de là, si on demande au système de chercher « livre », il recherchera un de ces deux objets. En revanche, si on lui demande de chercher « livre vert » seul le premier objet sera recherché.

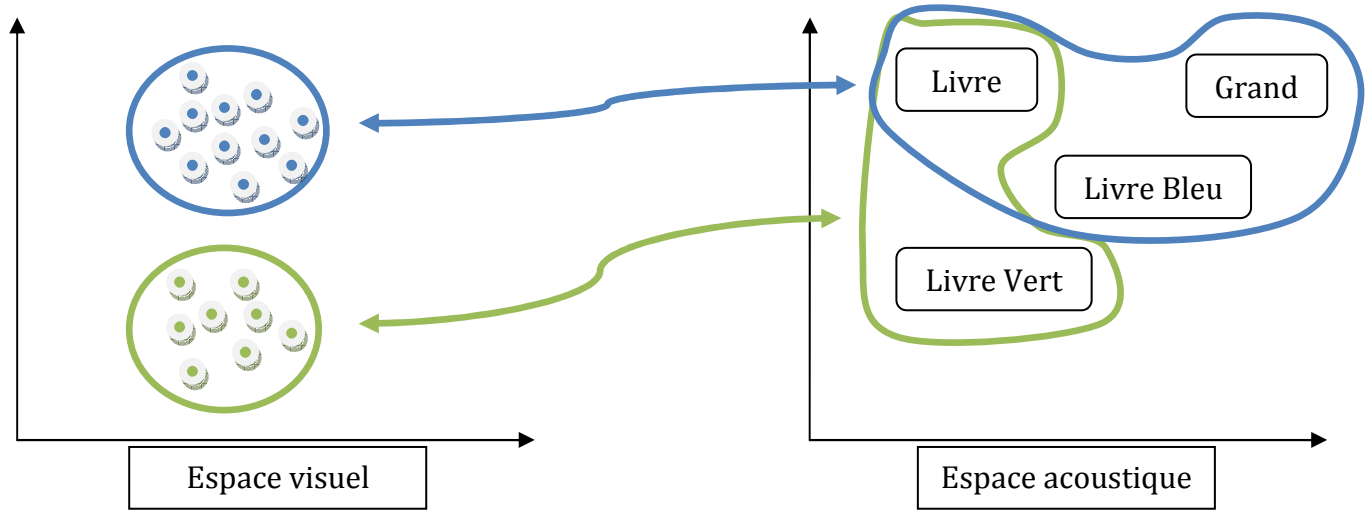


Figure 20 : Correspondance espaces visuel/acoustique

2.4.3 Restitution

Une fois que le robot possède une base de connaissances, il peut ensuite les restituer. Pour cela, deux méthodes sont mises à la disposition de l'utilisateur. La première, est de demander au robot s'il connaît un objet présent devant sa caméra. Le robot prend une photo puis compare avec toute sa base et indique le résultat en utilisant la méthode de correspondance présentée précédemment.

La deuxième méthode consiste à « chercher » un ou plusieurs objets. L'utilisateur prononce un son, le système retrouve la catégorie et donc les objets correspondants. Puis, il va tester l'image issue de sa caméra avec les films des objets à trouver. Il ne teste pas tous les films de sa base, mais seulement les films des objets qui correspondent au son. Cela permet d'une part d'effectuer le traitement plus vite. Mais cela permet surtout de ne pas prêter attention aux autres objets. Par exemple, considérons le cas où l'on cherche un téléphone et que celui-ci est posé sur un livre connu. Si on le cherche en se reportant à toute la base, les connaissances sur le livre vont perturber la recherche du téléphone.

Durant la phase de recherche, plusieurs solutions sont possibles : l'image devant la caméra présente très peu de similarité avec les films de la base de connaissances, l'image présente quelques similarités ou le système pense qu'un des objets est présent sur l'image. Dans le premier cas, le système indique que rien n'est reconnu, la recherche doit alors continuer de façon

hasardeuse ou en utilisant un algorithme qui permet de parcourir l'environnement. En revanche, si un certain nombre de correspondances sont trouvées le système va indiquer la zone où il y a le plus de correspondances. Cela peut par exemple servir pour orienter les recherches. Imaginons que 18 correspondances sont faites avec la partie droite de l'image issue de la caméra. Il semblerait intéressant d'orienter les recherches de ce côté. Toutefois, pour réussir à indiquer ces zones d'intérêt nous sommes obligés d'utiliser un algorithme de regroupement. En effet, comme il a été dit précédemment, il est possible que des liens soient faits avec le décor. Dans ce cas, on ne peut pas se contenter d'entourer la zone couvrant tous les points, il faut trouver la zone à plus forte densité.

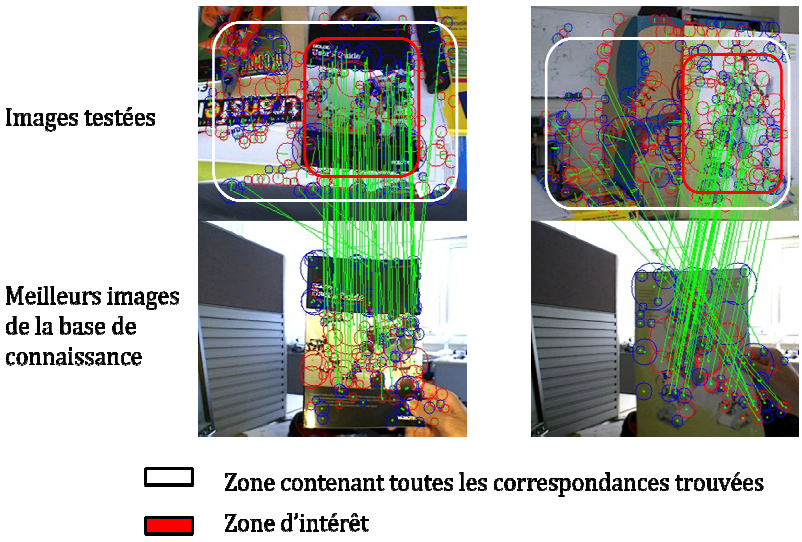


Figure 21 : Zone d'intérêt

Pour répondre à ce problème nous utilisons une méthode assimilable à un algorithme de « *clustering* » hiérarchique qui va regrouper les points en fonction de leur proximité. Dans un premier temps, nous calculons la distance minimale moyenne pour tous les points. Pour cela, nous parcourons tous les points de l'image et nous calculons la distance par rapport au voisin le plus proche. Puis, nous faisons la moyenne de cette distance pour tous les points. Enfin, on regroupe les points grâce à cette distance. Un point appartient à un groupe s'il est à une distance inférieure à D fois la distance minimale moyenne d'un des points du groupe. D'après nos essais, D doit avoir une valeur comprise entre 2 et 3.

```
Procedure regroupe

    # ce vecteur représente la distance du point le plus proche
    # pour tous les points de l'image 1
    minDistance[ points1.taille() ];
    # on trouve la distance du plus proche pour chaque point
    pour( i de 0 à points1.taille() )
        # on ne fait le calcul que si le point correspond avec l'image de la base
        si( points1[i].existCorrespondance() )

            pour( j de i + 1 à points1.taille() ){

                si( points1[j].existCorrespondance()  ){

                    dist = distance(points1[i] , points1[j] ));

                    # si cette distance est la meilleur pour le point 'i'
                    si( dist < minDistance[i].dist ){
```

```
                        minDistance[i].dist  = dist;
                        minDistance[i].index = j;
                    }
                    # si cette distance est la meilleur pour le point 'j'
                    si( dist < minDist[j].first ){
                        minDistance[j].dist  = dist;
                        minDistance[j].index = i;
                    }

                }
            }
        }
    }

    # on calcul la moyenne de cette distance du plus proche voisin
    distMinMoy = 0;
    nb_correspondance = 0;
    pour( i de 0 à points1.taille() ){
        si( points1[i].existcorrespondance() ){
            distMinMoy += minDistance[i].dist;
        }
    }
    distMinMoy = distMinMoy / nb_correspondance;


    # vecteur pour savoir si un point est déjà dans un groupe
    est_dans_groupe[ points.taille() ];
    # on parcourt tous les points
    pour( i = 0 à points1.taille() ){
        # si le point a une correspondance et qu'il n'est pas déjà dans un groupe
        si( points1[i].existcorrespondance() ET ! est_dans_groupe[i] ){

            # alors on créer un nouveau groupe
            Nouveau_groupe.creer() ;
            Nouveau_groupe.ajouter( points1[i] );
            # on indique qu'un le point est maintenant dans un groupe
            est_dans_groupe[i] = VRAI;

            # ce booléen indique que qu'un point a été ajouté au groupe
            # ce qui implique qu'il faut tester de nouveau tous les points
            refaire_un_tour = VRAI;

            tant_que(refaire_un_tour){

                refaire_un_tour = FAUX;
                # pour tous les points
                pour( j de 0 à points1.taille()){
                    # si le point à une correspondance et qu'il n'est pas déjà dans un groupe
                    # et qu'il est a une distance inférieure à X*distMoy du groupe
                    si( ! est_dans_groupe[j] ET points1[j].existcorrespondance()
                            ET appartient(Nouveau_groupe, points1[j] , X * distMinMoy) ET i != j ){
                        refaire_un_tour = VRAI;
                        Nouveau_groupe.ajouter( points1[j] );
                        est_dans_groupe[j] = VRAI;
                    }
                }

            }

            Groupes.ajouter(Nouveau_groupe) ;

        }
    }
}
```

Algorithme 6 : Regroupement des points en utilisant la distance moyenne du plus proche voisin

De cette manière, on va pouvoir trouver dans l'image la zone avec la plus grande densité de correspondances. Pour rajouter de la robustesse, le robot indiquera qu'il a trouvé l'objet s'il le trouve sur deux images successives. En effet, la probabilité de se tromper deux fois de suite est très faible.

2.5 Aspects techniques

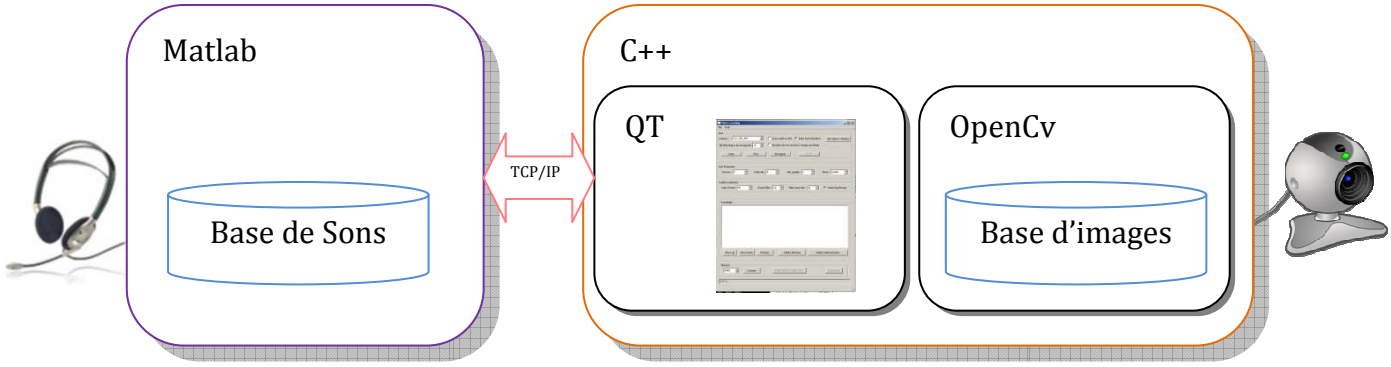


Figure 22 : Structure du système

La figure précédente montre la structure du système. Le module audio est développé en Matlab et communique avec le module vidéo grâce à un système de socket (TCP/IP). Le module vidéo est développé en C++. Il comporte une partie traitement de l'image qui repose sur OpenCv et une partie interface graphique et communication réseau basée sur QT.

Cette structure permet d'exécuter les deux modules sur des processeurs différents ou des ordinateurs différents. Ce qui peut être avantageux de par le besoin en ressource de chaque module. L'échange entre les deux modules est minime, il consiste en quelques messages qui indiquent l'état du système et les évènements (moins de 10 messages de moins de 20 caractères) il n'était donc pas intéressant d'utiliser la programmation parallèle.

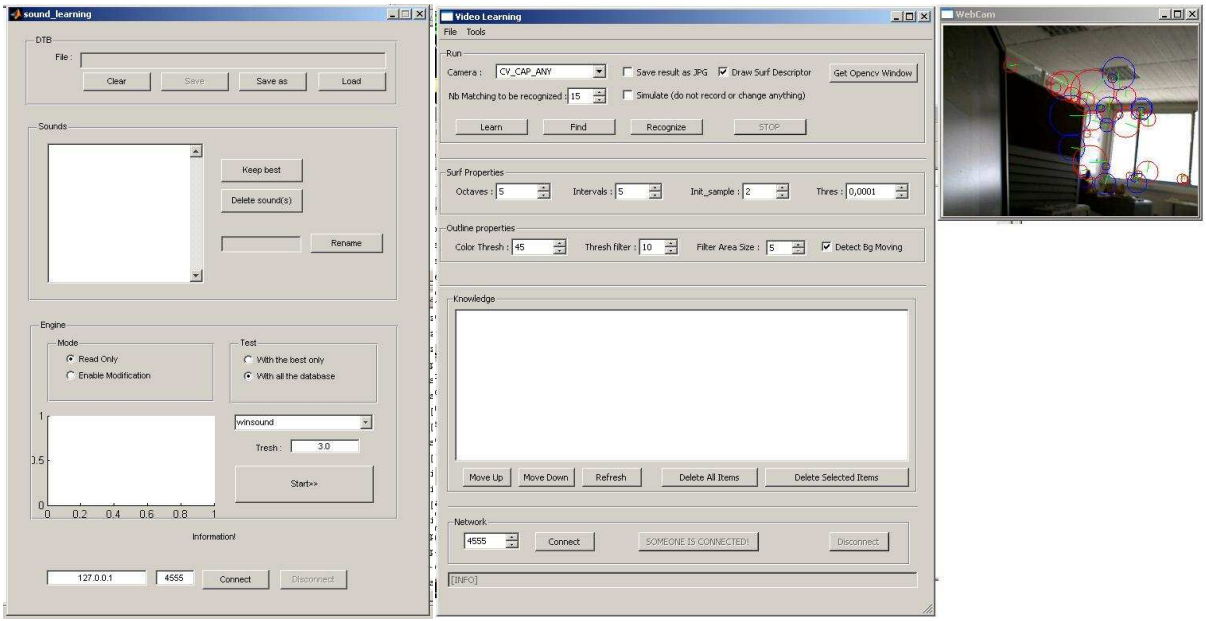


Figure 23 : Interfaces Matlab – Qt – OpenCv

2.6 Utilisation sur humanoïde

2.6.1 Introduction

Après avoir créé le système, nous avons mis en application le système sur Nao qui est un robot humanoïde. Cela permet de savoir si les interactions sont faciles à réaliser et de savoir si un robot pourrait retrouver les objets.

2.6.2 Nao

- **Présentation**

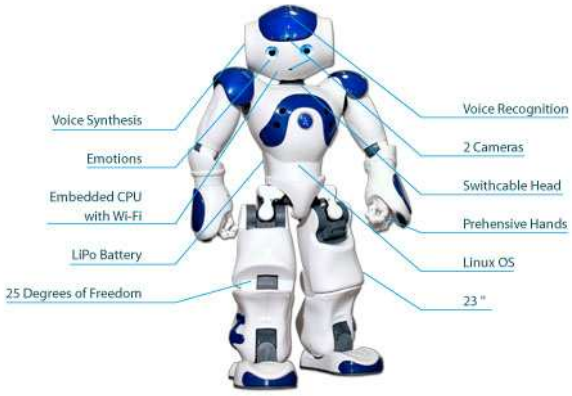


Figure 24 : Nao

Nao est un robot de type humanoïde (à forme humaine) créé par Aldebaran une entreprise française. Ce robot de 60 cm est équipé de nombreux capteurs et outils. Toutefois, ce robot n'est à l'origine doté d'aucune intelligence. Mais il est possible de manipuler tous les composants : moteurs, LEDs, parole, etc pour lui donner le comportement voulu.

- **Interaction hommes-machines**

L'objectif premier de cette mise en application est de savoir si il serait facile pour un non spécialiste d'utiliser ce système : enseigner les objets, tester les connaissances, etc.

Cette approche implique de se passer des informations accessibles sur l'écran. En effet, il est possible de voir l'état du système et ce que voit le robot grâce aux interfaces présentes sur l'écran de l'ordinateur. Mais si le système est utilisé de façon autonome sur un robot il n'y a pas d'écran, il faut donc trouver d'autres moyens pour indiquer l'état du robot et de l'apprentissage auprès de l'utilisateur.

Pour cela nous utilisons les LEDs ainsi que différentes position qui permettent de montrer à l'utilisateur dans quel état est le robot. Ces positions se doivent d'être communes, par exemple lorsque le robot ne peux pas apprendre – car le fond est en mouvement – il va placer les bras devant le buste et allumer les LEDs rouges ce qui peut facilement être interprété.

Normal

Prêt à apprendre

Ne peut pas apprendre

En train d'apprendre

Cherche un objet

A trouvé un objet (Leds clignotantes)

Figure 25 : Position clés de Nao pour s'exprimer

2.6.3 URBI

Aldebaran propose pour les développeurs d'applications sur le robot Nao, des librairies C++ ou Python. En les utilisant, les applications sont alors directement exécutées sur le robot. Cela représente un avantage important lorsque l'on cherche à rendre celui-ci autonome et pour des applications consommant peu de ressources. Néanmoins, dans notre situation, il paraît intéressant de laisser des informations affichées sur l'écran de l'ordinateur, comme par exemple les images perçues par le robot ou les informations apprises. Utiliser les outils propres à Nao ne semble pas convenir complètement. De plus, si par la suite on souhaite utiliser l'application sur un autre robot cela nécessiterait de modifier une partie importante du programme.

C'est en partie pour ces raisons que des développeurs ont créé URBI. Développé par GOSTAI, URBI (*Universal Real-Time Behavior Interface*) est un *framework* et un langage qui a pour objectif d'ajouter une couche d'abstraction pour le développement dans la robotique ludique ainsi que de créer un langage compatible sur tous les robots.

- **Le langage**

Le langage script URBI possède une syntaxe mélangeant C++ et Ruby. Un travail important a été réalisé pour la gestion du parallélisme et des évènements. Des scripts URBI peuvent être directement exécutés dans un terminal pour contrôler un robot. Comme le montre la figure suivante qui illustre certaines actions que nous utilisons sur Nao. Il faut comprendre que ces lignes sont interprétées dès qu'elles sont écrites dans le terminal.

```
# Allume les moteurs pour leur permettre de bouger
motors.on();

# Déclare une variable 'stiffness' qui a pour valeur 0.5
var stiffness = 0.5;

# Règle la dureté des moteurs de l'épaule droite
RShoulderPitch.setStiffness(stiffness);
RShoulderRoll.setStiffness(stiffness);

# Déclare une fonction (sans l'exécuter) qui allumera les
# leds en boucle vert/bleu/rouge
var mytagled = Tag.new();
function loopLeds{
    mytagled : {
        loop{
            greenLeds();
            sleep( 0.3s );
            blueLeds();
            sleep( 0.3s );
            redLeds();
            sleep( 0.3s )
        }
    },
};

# Exécute la fonction et attend 5s en même temps
loopLeds(),
sleep( 5s );
# Stop la boucle
mytagled.stop();

# Fait bouger la tête à droite en 2s et dis "hello"
# en même temps
```

```
headYaw.val = 1 smooth:2s,
tts.say("Hello");
```

Algorithme 7 : Exemple de script URBI sur Nao (Code)

Nous avons créé un ensemble de fonctions et de mouvements clés avec Nao comme illustré dans la figure précédente puis, il a fallu les inclure dans notre système déjà existant.

- **Intégration**

URBI est fourni avec des logiciels et des librairies permettant d'exécuter les scripts directement dans le PC embarqué d'un robot ou à distance en utilisant le réseau (WIFI, RJ45, Bluetooth, etc.). Dans le premier cas, une librairie C++ permet d'utiliser URBI en faisant abstraction de l'architecture du système i.e. de la connexion et des scripts. Cette solution aurait été idéale. Toutefois, depuis la version 2.0 d'URBI, la librairie d'utilisation C++ n'est pas compatible avec MINGW. En effet, bien que multiplateformes, la version 2.0 n'est compatible qu'avec le compilateur propriétaire de Microsoft sous Windows. Or, tout notre système utilise MinGW pour créer des applications par la suite ensuite compatible avec Linux et Mac OS X.

Pour répondre à ce problème nous avons utilisé une solution simple qui consiste à se connecter en TCP/IP sur le serveur URBI de Nao et d'envoyer le script directement sur le réseau. Ainsi, au démarrage de l'application, nous envoyons sur le réseau un fichier d'initialisations contenant la déclaration des variables et des fonctions. Puis, lors de l'exécution nous envoyons une trame simple, en fonction des évènements, qui n'est rien d'autre qu'un appel à une des fonctions chargées. Par exemple, lorsque le robot est prêt à apprendre nous envoyons ''readyToLearn() ;\n''.

Cependant, cette solution a, par la même occasion, amené un nouveau problème. En effet, l'avantage d'utiliser la librairie URBI est d'avoir accès à tous les contrôles d'un robot y compris la caméra. En communicant directement avec le serveur URBI il faut alors demander manuellement une image qui nous est retourné au format JPEG mais sous forme de tram TCP/IP. Il s'agit alors de transformer manuellement un buffer contenant une image JPEG en tableau de pixels BGR exploitable par notre système. Bien que ce travail soit en partie réalisé grâce à la librairie libjpg cela pose des problèmes de latence et d'accès à une vidéo en réseau : image incomplète, image corrompue, latence due au réseau ou surcharge du CPU du robot.

2.6.4 Structure du système

L'ajout d'URBI dans notre système modifie son architecture puisqu'il est nécessaire de récupérer le flux vidéo provenant de Nao, le traiter et renvoyer des informations à Nao pour que celui-ci prennent les positions voulues. De cette manière, l'utilisateur à l'impression que le robot autonome mais le traitement est délégué à un PC tiers qui peut dans un même temps afficher des informations.

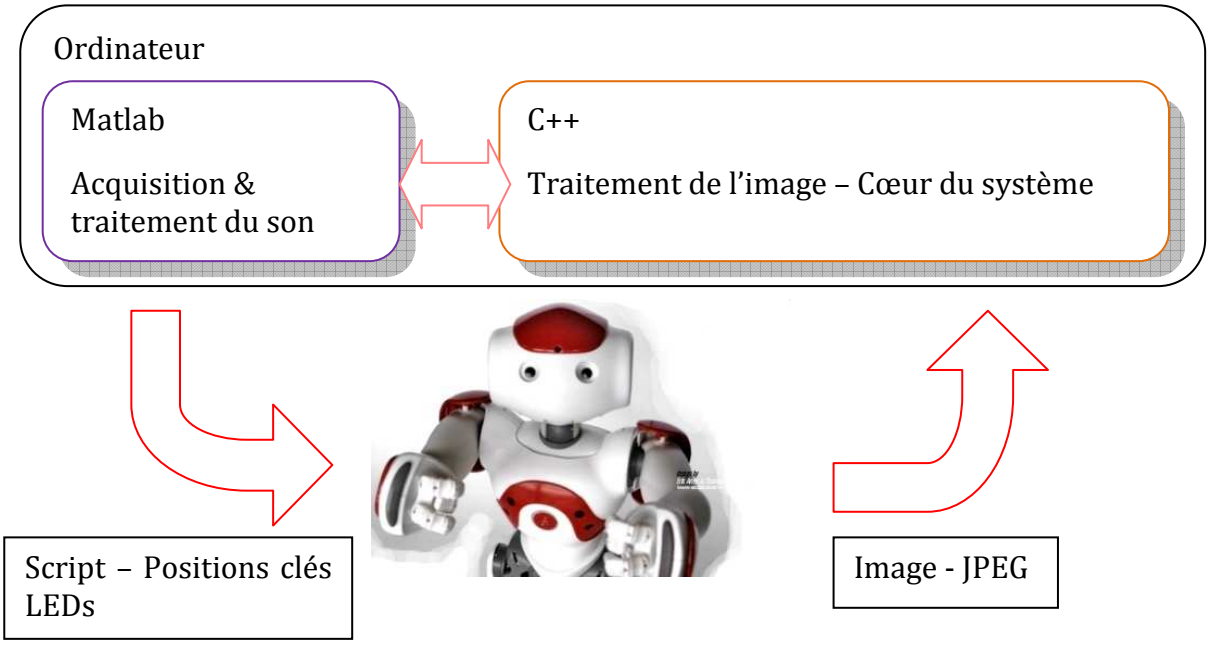


Figure 26 : Structure du système avec Nao et URBI

2.7 Limites de l'architecture

L'architecture du système est relativement complexe. Il parait difficile d'envisager mettre un tel système directement dans l'ordinateur d'un robot. En effet, les temps de calcul risque d'être beaucoup plus long et la mémoire vive nécessaire au chargement des données pour les descripteurs SURF des films est importante (plusieurs centaines de méga octets). Or, les systèmes présents sur les robots ludiques sont très contraignants.

De plus, bien que tout ce qui ait été développé soit multiplateforme, le mélange de toutes ces technologies en fait un système difficile à faire évoluer.

3 Discussion et perspective

3.1 Organisation du stage

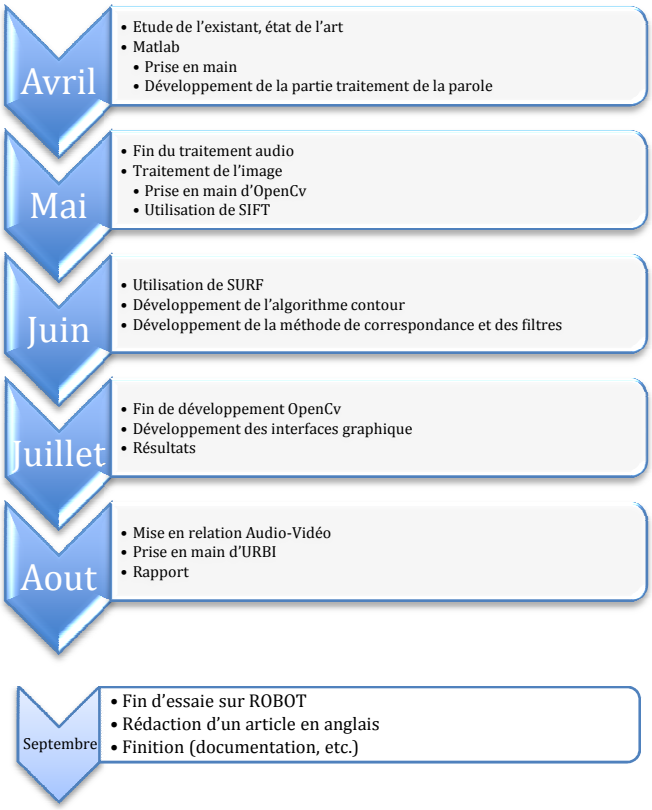


Figure 27 : Organisation du stage

Dans un premier temps, j'ai effectué une étude des articles et ouvrage traitant de sujets proches du mien. Cet aspect du travail est à mettre en relation avec l'orientation recherche de mon stage. La plupart des articles étudiés ont été présentés dans la partie « Travaux connexes ».

Puis, j'ai réalisé le développement du traitement de la voix à l'aide de Matlab. Cela m'a demandé dans un premier temps une mise à niveau dans l'utilisation de Matlab.

Ensuite, j'ai travaillé sur l'aspect vidéo et reconnaissance. Durant cette phase de travail, j'ai pris en main *OpenCV*, j'ai réalisé des essais à l'aide de libraire SIFT puis je me suis consacré à l'utilisation de *OpenSURF*.

Enfin, la dernière partie de mon travail a consisté à effectuer des tests, unifier les deux modules, développer des interfaces graphiques et utiliser Nao.

Ce rapport a été rédigé alors qu'il me restait plus d'un mois de travail. C'est pour cette raison que des activités ont été laissées pour le mois de septembre sans avoir été présentées dans ce rapport.

3.2 Perspectives

Les perspectives sont diverses et vont de l'utilisation possible de ce système aux progrès et changement d'un point de vue algorithmique.

- **Modifications potentielles**

D'après nos essais, SIFT donnait davantage de descripteurs dans une image mais avait besoin de beaucoup plus de temps que SURF pour fournir des résultats. Il serait néanmoins intéressant d'utiliser SIFT dans notre système en reprenant les mêmes algorithmes et les mêmes données pour comparer les résultats.

Lorsque nous comparons deux images, le résultat des comparaisons entre deux descripteurs est au dessus du seuil dans la majorité des cas. Or, dans des approches gloable, comme l'approche « sac de mot », il est possible de trier les descripteurs avec des algorithmes comme KMEANS. Mais dans notre système cela n'est pas possible puisqu'il faut que nous gardions des informations relatives à l'image et à la position du descripteur par rapport aux autres pour le filtre spatial. Il faudrait sans doute réfléchir de manière à faire des KMEANS au sein des images de cette manière on n'aurait pas à comparer tous les descripteurs entre eux.

Le filtre spatial, présenté dans ce rapport, fait que pour un descripteur on récupère les informations de ses voisins. Or, dans notre filtre angulaire nous prenons l'information angulaire globale. Et cela peut poser des problèmes. Par exemple, si on enseigne au système un objet composé en plusieurs parties amovibles les unes aux autres. Si on teste ensuite la connaissance de cet objet mais que l'on a changé l'orientation que d'une seule partie de l'objet, le filtre angulaire va agir brutalement et supprimer de nombreux bons descripteurs. Il paraît alors judicieux d'utiliser un filtre angulaire local (au niveau des voisins) comme le filtre spatial.

Les traitements pourraient être parallélisés. En effet, puisque on compare tous les sons et toutes les images un à un, il serait possible de le faire en parallèle. Toutefois, peu de robots sont équipés de multiprocesseurs. Il faudrait alors obligatoirement déléguer le traitement sur un ordinateur.

Dans ce rapport, nous avons présenté des algorithmes pour savoir si un objet est dans une image et pour entourer la zone à forte densité de correspondance dans cette image. Mais nous n'avons pas travaillé sur la possibilité que plusieurs objets connus soit dans la même image. Pourtant, ca ne serait qu'une extension des algorithmes utilisés. On pourrait imaginer que si plusieurs objets sont présents dans une image chacun d'entre eux aurait une zone d'intérêt et donc une position.

D'après nos résultats, les objets planaires sont beaucoup plus reconnaissables par notre système que les autres. Il semblerait intéressant de chercher à savoir d'où vient cette différence et s'il n'est pas possible de trouver une solution.

- **Utilisation**

Ce système peut être utilisé dans de nombreuses situations : HRI, étude comportementale, etc. Par exemple, il parait intéressant de faire des tests avec des personnes non initiées pour voir si elles arrivent à enseigner des objets au robot ou non. En effet, bien que le système paraisse intuitif, le faire fonctionner en situation réelle avec un public non informaticien risque de faire apparaître des problèmes et des défauts.

Le système étant robuste et complet il serait tout à fait imaginable de le commercialiser pour des robots ludiques. Comme il a été expliqué dans la partie architecture, cela demanderait à déléguer une partie importante du travail sur un ordinateur et de ne pas utiliser Matlab. Mais en utilisant URBI, de nombreux robots seraient facilement compatibles.

Conclusion

Le travail effectué durant ce stage avait pour objectif la réalisation d'un système d'apprentissage pour robot. Les différents objectifs fixés au début ont été atteints. Le système peut apprendre des objets en temps réel et sans le moindre pré-requis. La reconnaissance visuelle des objets, qui a demandé la création de nouveaux algorithmes, donne de très bons résultats pour les objets plats. De plus, la méthode d'enseignement est très intuitive puisqu'il suffit d'agiter l'objet devant la caméra en prononçant des sons pour que le robot élargisse sa base de connaissances. Ce type d'interaction peut être fait par des personnes non spécialistes et se rapproche fortement de geste intuitif que font des adultes avec des nouveaux nés.

L'intérêt de ce système est de permettre à n'importe quel robot possédant une caméra d'apprendre des objets. De cette manière, il peut partager des informations sur le monde avec l'utilisateur qui pourrait par exemple lui demander de retrouver un objet particulier dans une pièce.

Des améliorations peuvent apporter au niveau de la structure informatique du système ainsi que de la rapidité. En effet, sur un ordinateur le programme s'exécute facilement, mais il serait difficile à exécuter sur les architectures présentes au sein des robots. De plus, une amélioration peut être apportée au niveau de la comparaison des descripteurs SURF. Par exemple, la méthode « sac de mots » permet d'utiliser des algorithmes de tris comme K-MEANS. Chercher une approche similaire dans notre système en triant ou en regroupant les descripteurs SURF pour ne pas avoir à faire de comparaisons inutiles permettrait de gagner énormément de temps.

Les problèmes rencontrés durant ce travail sont divers. Au niveau informatique, il a fallu faire des choix sur les langages ou les librairies à utiliser pour dès fois ce rendre compte que ces choix n'étaient pas optimaux. Enfin, l'aspect recherche de ce stage m'a demandé un travail personnel particulier car je n'avais pas d'expériences similaires.

Les perspectives du travail effectué sont nombreuses. Il semblerait intéressant de faire un logiciel commercial utilisant les résultats de ce stage. Mais les résultats peuvent aussi donner lieu à de nouvelles recherches au niveau expérimental, comme par exemple pour la mise en correspondance des descripteurs SURF ou encore la formalisation des interactions entre robots et humains.

Bibliographie

Chen, Y., & Dana, H. B. (2004). On the integration of grounding language and learning objects. *AAAI-2004* , 488–493.

CNRTL. (s.d.). Consulté le 2009, sur Centre National de Ressources Textuelles et Lexicales: http://www.cnrtl.fr/lexicographie/intelligence

Daniel, P., & W., E. (2005). *PLP and RASTA (and MFCC, and inversion) in Matlab.* Récupéré sur http://www.ee.columbia.edu/~dpwe/resources/matlab/rastamat/

Deb, R., Bernt, S., & Alex, P. (1999). Learning Audio-Visual Associations using Mutual Information. *Integration of Speech and Image Understanding* , 147.

FILLIAT, D. (2007). A visual bag of words method for interactive qualitative localization and mapping. *ICRA (Internationnal Conference on Robotics and Automation).* Kobe, Japan.

Frank, L., & Gerhard, S. (2002). A Multimodal System for Object Learning. *Lecture Notes in Computer Science 2449* , 490–497.

Gabriella, C., Christopher, R. D., Lixin, F., Jutta, W., & Cédric, B. (2004). *Visual Categorization with Bags of Keypoints.* Xerox Research Centre Europe.

H., H., & N., M. (1994). RASTA processing of speech. *IEEE Trans. on Speech and Audio Proc., vol. 2, no. 4* , 578-589.

Heiko, W., Stephan, K., Michael, G., Holger, B., Mark, D., Inna, M., et al. (2006). A Biologically Motivated System for Unconstrained Online Learning of Visual Objects. *ICANN (International Conference on Artificial Neural Networks)* (pp. 508–517). Springer.

Herbert, B., Andreas, E., Tinne, u., & Luc, V. G. (2008). SURF: Speeded Up Robust Features. *Computer Vision and Image Understanding (CVIU), Vol. 110, No. 3* , 346-359.

Herbert, B., Tinne, T., & Luc, V. G. (2006). SURF: Speeded Up Robust Features. *9th European Conference on Computer Vision.* Graz.

Hermansky, H. (1990). Perceptual linear predictive (PLP) analysis of speech. *J. Acoust. Soc. Am., vol. 87, no. 4* , 1738-1752.

Iwahashi, N. (2004). Active and Unsupervised Learning for Spoken Word Acquisition Through a Multimodal Interface. *IEEE International Workshop on Robot and Human Interactive Communication.* Okayama Japan: 437-442.

Josef, S., & Andrew, Z. (2008). Efficient Visual Search for Objects in Videos. *Proceedings of the IEEE, Vol. 96, No. 4* , 548-566.

Kevin, M. S., & Stephen, E. L. (2005). HMM-Based Semantic Learning for a Mobile Robot. *Transactions on Evolutionary Computation , VOL. 11* , 154.

Kobus, B., Pinar, D., David, F., Nando, d. F., David, M. B., & Michael, I. J. (2003). Matching Words and Pictures. *Journal of Machine Learning Research 3* , 1107–1135.

L. R., R., & B., J. (1993). *Fundamentals of speech recognition (Chapitre 4).* Prentice-Hall, Inc.

Lopes, L. S., & Chauhan, A. (2007). How many words can my robot learn? *Interaction Studies 8:1* .

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

Lowe, D. G. (1999). Object recognition from local scale-invariant features. *Proceedings of the International Conference on Computer Vision 2* , 1150–1157.

OKA, N., NAKAFUSHIKI, A., & ITOH, Y. (2005). Learning the Correspondence between Continuous Speeches and Motions. *IEEE International Conference on Development and Learning* .

Roy, D. (2005). Grounding words in perception and action: computational insights. *TRENDS in Cognitive Sciences Vol.9 No.8* .

Steels, L., & Kaplan, F. (2000). AIBO's First Words. The Social Learning of Language and Meaning. *Evolution of Communication* , 3-32.

Annexes

- Sites en relation avec des éléments du stage :

FLOWERS : http://flowers.inria.fr/

GOSTAI/URBI : http://www.gostai.com/

Aldebaran : http://aldebaran-robotics.com/

OpenCV : http://sourceforge.net/projects/opencvlibrary/

- Détaille des traitements PLP-Rasta : Rast-PLP Speech analysis, Hynek H., Nelson M., Aruna B. & Phil K., 19991 (http://www.icsi.berkeley.edu/pubs/techreports/tr-91-069.pdf)

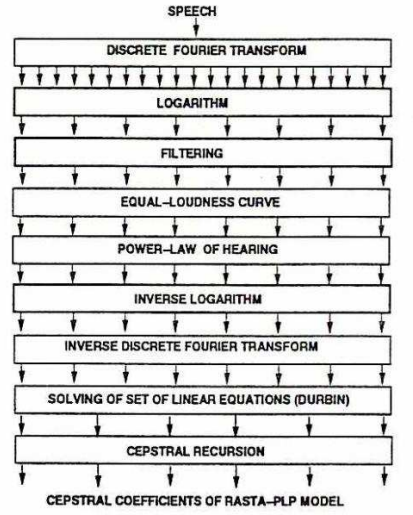


Figure 1: RASTA-PLP Method