Rémi RIGAL
CI2018 - Profil SPID/ROB

UV 6.2 - Rapport PFE

Mise en œuvre d'un dialogue multimodal avec un robot

Tuteur Entreprise
Jacques Chodorowski

Tuteur Ecole
Benoit Zerr

27 août 2018

Remerciements

Avant de débuter la rédaction de ce rapport de projet de fin d'études, je souhaiterais remercier plusieurs organisations et personnes.

J'adresse mes sincères remerciements à l'ENSTA Bretagne pour le cadre pédagogique qu'elle offre avec des enseignements de qualité adaptés aux besoins des entreprises, mais aussi des enseignants motivés et intéressants très à l'écoute des élèves. Ce cursus m'a permis d'avoir un niveau de connaissance me permettant de construire une future carrière à la hauteur de mes exigences.

Je souhaite également remercier l'entreprise Orange pour le cadre de travail dont j'ai disposé tout au long de mon stage, un cadre au sein duquel j'ai eu l'occasion de m'épanouir à la fois personnellement et professionnellement. Cette première immersion dans le monde du travail a été idéale pour me préparer à la suite de ma carrière.

Je remercie toutes les personnes de l'équipe dans laquelle j'ai travaillé : l'équipe HFC, pour leur accueil chaleureux et la bonne humeur quotidienne qui domine dans l'équipe.

J'aimerais également adresser des remerciements particuliers à plusieurs personnes sans qui je n'aurais pas eu les mêmes opportunités.

Mr Benoit Zerr, mon tuteur école, pour m'avoir accompagné et soutenu dans mes projets durant ces trois années et pour son enthousiasme communicatif qui font de lui un excellent enseignant.

Mr Jacques Chodorowski, mon tuteur entreprise, que je remercie tout particulièrement pour la confiance qu'il m'a accordé ainsi que pour son apport technique. Je le remercie également pour le temps et l'énergie dépensés pour construire mon futur au sein d'Orange.

Pour finir je souhaite sincèrement remercier toutes les personnes ayant participé de près ou de loin à la réussite de mes études.

Résumé

Consciemment ou non, de nombreuses informations sont communiquées, perçues et échangées lors d'une conversation entre deux humains. Elles viennent enrichir l'interaction en proposant des interprétations différentes, des dimensions qu'un dialogue textuel ne peut saisir. Ces signaux sont souvent difficiles à qualifier tant ils sont le produit des émotions humaines, éphémères et propres à chacun. Il est pourtant essentiel de pouvoir les interpréter et les prendre en compte dès lors que l'on veut mettre en place une interaction avec un humain.

Dans cet optique, la dimension principale de ce stage est la mise en œuvre d'un système de dialogue multimodal pour un robot, chaque mode étant un flux d'information pouvant être capté ou communiqué. L'objectif est donc de mettre à profit l'ensemble des capteurs et actionneurs que peut posséder une telle plateforme afin de construire un dialogue avec un humain. Ce projet est en accord avec l'émergence et la popularisation des systèmes multi-capteurs et multi-actionneurs que l'on peut constater aujourd'hui tels que les smartphones ou les enceintes connectées.

Le moteur de dialogue multimodal développé au cours de mon stage répond aux attentes et au cadre défini. Cependant bien que parfaitement fonctionnel il trouve ses limites dans son manque d'adaptabilité vis à vis de l'environnement dans lequel il évolue. De plus la machine à états finis régissant la logique des échanges rend ces derniers très déterministes et donc peu naturels. Dans ce contexte il apparaît pertinent de poursuivre mes travaux en améliorant ces deux points afin de proposer un dialogue au plus près des interactions humaines.

Abstract

Consciously or not, a huge amount of data is transmitted, perceived and exchanged during a conversation between two human beings. They enrich the interaction by allowing a large diversity of interpretations and create dimensions that a textual-only dialogue cannot grasp. Those data are often hard to qualify as they are the outcome of ephemeral and subjective human emotions. However, being able to interpret them and take them into account is fundamental for anyone who wants to implement a human-machine interaction.

In this perspective, the main goal of this traineeship is the implementation of a multimodal dialogue for a robot, every modality being a stream of data that can be perceived or emitted. Therefore, the intention is to take advantage of the set of sensors and actuators present on such a device in order to implement an as complete as possible interaction with a human being. This project is relevant as it finds many usages in our everyday life multi-sensors and multi-actuators devices such as smartphones or connected speakers.

Ultimately, the multimodal dialogue engine developed during my traineeship met the expectations. However, while being perfectly functional it is limited by its lack of adaptability regarding the environment in which the system evolves. Moreover, the finite state machine used for the dialogue logic makes the discussion very determinist and therefore less natural. In this context, it might be interesting to improve those two points in order to achieve a complete human-like interaction.

Table des matières

Introduction

Les travaux présentés dans ce document ont été réalisé dans le cadre de mon projet de fin d'études (PFE) au sein d'Orange sous la tutelle de Jacques Chodorowski. Ce projet est l'aboutissement de cinq années d'études dont deux en classes préparatoires et trois en cycle ingénieur à l'ENSTA Bretagne. L'objectif de ce stage est de mettre en œuvre un dialogue multimodal avec un robot, ce projet rentre donc en parfaite adéquation avec mon cursus spécialisé dans la robotique. Mon choix a été de plus motivé par le côté innovant du sujet centré sur une problématique actuelle au cœur d'un domaine en plein essor.

J'ai effectué mon stage au sein de l'entreprise Orange qui est aujourd'hui l'un des principaux acteurs de la télécommunication en Europe. Elle tire son épingle du jeu en valorisant la recherche et l'innovation par le biais de ses Orange Labs que j'ai eu la chance d'intégrer. Au cœur d'un projet de recherche en robotique, mes missions sont non seulement l'étude et la prospection mais aussi le développement de service.

En premier lieu est présenté dans ce rapport un aperçu du contexte de mon stage, l'entreprise Orange ainsi que l'environnement de travail dans lequel j'ai évolué. Ensuite suit une présentation en détails de mon sujet de PFE avec sa dimension, son contexte et ses objectifs. Par la suite un état de l'art sur la multimodalité appliquée aux dialogues détaille ses utilisations actuelles, et plus particulièrement dans le domaine de la robotique. Finalement sont détaillés mes travaux, réalisés d'une part sur le moteur de dialogue et d'autre part en collaboration avec les autres membres du projet robotique.

Chapitre 1

Contexte

1.1 Orange

1.1.1 Entreprise Orange

Orange est une entreprise française dont l'histoire remonte à 1889 avec la création du ministère des P&T [1]. Elle deviendra plus tard la société anonyme France Telecom puis prendra finalement le nom d'Orange à partir de 2001. Elle a le statut d'opérateur public historique et l'état français est encore aujourd'hui actionnaire majoritaire de la société. Elle a de plus su tirer parti de ses compétences pour obtenir une dimension internationale forte, notamment dans les pays de la zone Moyen-Orient et Afrique (MEA).

Historiquement les secteurs d'activités d'Orange sont la téléphonie fixe et mobile, internet et la vente. Depuis peu elle a développé des services bancaires comme Orange Money [2] ou Orange Cash [3] tout deux très populaires dans les pays de la zone MEA. Finalement l'entreprise a lancé il y a peu le service de banque mobile Orange Bank afin de conforter sa position dans les activités bancaires. On retrouve également des activités annexes telle que la télévision avec OCS [4], la musique avec Deezer, la marine avec les navires câbliers.

Toutes ces activités sont réparties au sein de trois entités différentes :

- **Le service commercial** qui assure la vente et la promotion du groupe à travers les boutiques ainsi que toutes les personnes travaillant dans des équipes à vocation business
- **Le service technique** qui gère la maintenance du réseau et son expansion
- **Le service R&D** qui a pour objectif de concevoir et créer les nouveaux services pour les clients

1. P&T : Postes et Télécommunications
2. Orange Money : Service permettant le transfert de l'argent sécurisé d'un mobile à l'autre
3. Orange Cash : Service de paiement mobile sans contact
4. OCS : Acronyme d'Orange Cinéma Série. Bouquet de chaînes de télévision françaises consacrées aux séries et au cinéma

Forte de ses 270 millions de clients et 152 000 employés [5], Orange est classée en 2017 à la 51ème place du classement des marques mondiales selon le Brand Finance et est actuellement la 4ème entreprise mondiale du secteur des télécommunications. Son chiffre d'affaires s'élève à 41,1 milliards d'euros pour un résultat net de 1,9 milliards d'euros [6]. Leader ou second opérateur dans 75% des pays européens où elle est implantée et dans 83% des pays en Afrique et Moyen-Orient, Orange a su s'adapter à des marchés internationaux différents. De plus, Orange a su convaincre avec sa qualité de service et son service client qui sont les deux fleurons de la marque.

Présente tout autour du globe, Orange doit faire face à une clientèle très hétérogène dont les attentes et besoins sont variés. D'une part 30 pays principalement d'Europe et d'Afrique portent le marché de l'offre grand public. D'autre part le reste du monde, soit 220 pays, constituent la clientèle professionnelle pour un total de 3000 multinationales clientes. La répartition du chiffre d'affaires peut elle aussi s'effectuer par zone géographique. En effet l'Europe représente 64,1% du CA, l'Afrique et le Moyen-Orient 18,29%, et le reste du monde 15,1%. La répartition de la clientèle du groupe est en accord avec celle du CA, en effet sur les 270 millions de clients dans le monde 211,4 millions sont des clients en téléphonie mobile et 19,5 millions des clients internet.

1.1.2 Orange Labs

Orange, classée 19ème au classement BCG [7] des entreprises les plus innovantes en 2017, a su faire de l'innovation une force reconnue internationalement. Plus de 5000 chercheurs, ingénieurs, techniciens et designers sont présents dans sa division R&D : les Orange Labs. Présents dans 12 Pays et avec 700 millions d'euros investis en 2018, soit 1,7% du chiffre d'affaires, les Orange Labs ont un portefeuille de 6498 brevets déposés.

Les Orange Labs sont une entité à part entière découpée en trois pôles, Orange Labs Services (OLS), Orange Labs Network (OLN) et Orange Labs Research (OLR). Ayant pour mission commune l'innovation, ces derniers sont respectivement dédiés au développement de services, au réseau de télécommunication et à la recherche.

1.2 Cadre Personnel de Travail

1.2.1 Équipe Administrative

Pendant mon stage j'ai travaillé avec l'équipe HFC, acronyme de Home Family Communication. Rattachée au pôle OLS, elle a pour mission de développer des services dédiés à l'amélioration des interactions interpersonnelles ou la création de services pour la maison. A titre d'exemple l'un des thèmes des travaux de l'équipe est la conception de la

5. Au 30 Septembre 2017
6. Au 31 Décembre 2017
7. BCG : Boston Consulting Group

future enceinte connectée Djingo.

1.2.2 Equipe Projet

L'équipe du projet robotique est composée de six personnes dont un chef de projet, un chercheur, deux apprentis en fin de cycle effectuant leur projet de fin d'études et deux stagiaires. Nos sujets de PFE traitent tous d'une partie indépendante mais permettent de constituer des cas d'usages divers et complets une fois mis en commun. Ces sujets sont respectivement :

- Détection de gestes par analyse d'images et apprentissage automatique
- Étude de la préhensibilité et des mouvements d'un bras robotisé à six degrés de liberté
- Navigation autonome et planification de tâches
- Mise en œuvre d'un dialogue multimodal

Nos travaux ont bénéficié de l'apport technique de Jacques Chodorowski qui supervisait nos travaux, ainsi que du support d'Yvan Picaud le responsable du projet.

1.2.3 Moyens à Disposition

Notre principal support de développement est le robot Waldo ci-contre. Il a été conçu par la startup française Immersive Robotics et est encore aujourd'hui en cours de développement.

Afin de capter au mieux son environnement il est muni d'un LIDAR, de télémètres à ultrasons, de deux caméras - une par oeil-, de microphones ainsi que d'une Kinect v2. Ces différents capteurs font de Waldo un robot polyvalent ayant une excellente perception de l'environnement dans lequel il évolue. Il est de plus muni de divers actionneurs comme une base motorisée, un bras à six degrés de liberté, une tête à deux degrés de liberté, des haut-parleurs et des LEDs de couleurs sur l'ensemble de sa tête. Cela lui permet d'interagir efficacement avec des interlocuteurs tout en lui garantissant une autonomie de déplacement. Finalement, son buste arbore une tablette Android permettant de décupler ses possibilités d'interactions. L'ensemble de ces composants offre de nombreuses possibilités d'usages, allant du robot de télé-présence au robot d'accueil interactif.

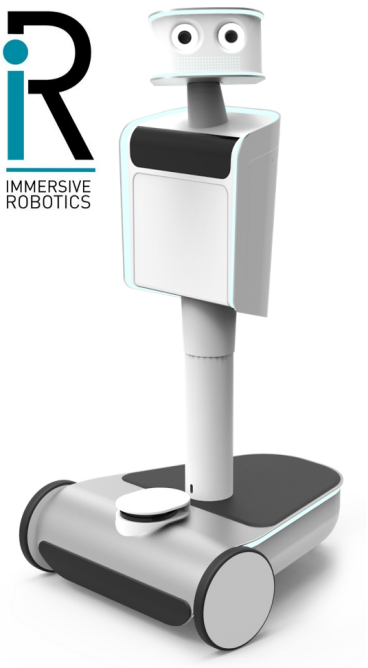

FIGURE 1.1 – Le robot Waldo

En termes de capacités de calculs Waldo est équipé dans sa version d'origine d'un micro-ordinateur Up Squared contenant un Intel Pentium ainsi que 8 Go de mémoire vive. Lors des essais nous avons constaté que celui-ci n'était pas suffisant et avons donc décidé d'ajouter un ordinateur Odroid équipé d'un processeur ARM Cortex-A53 et 2 Go de mémoire vive. En addition, une tour connectée au même réseau que le robot permet de déporter les calculs lourds afin d'alléger la charge des processeurs embarqués et ainsi bénéficier d'une réactivité optimale.

Chapitre 2

Dimension du Projet de Fin d'Études

2.1 Contexte

Des objets connectés de plus en plus complexes et performants arrivent dans la vie courante des particuliers et des entreprises. L'un des défis à relever est l'émission intuitive, sans apprentissage, de commandes vers ces objets selon un ou plusieurs modes de saisie mis à disposition. Les systèmes multimodaux, aujourd'hui peu nombreux, sont la réponse à la problématique des échanges complexes entre humains et robots. Ils offrent non seulement la possibilité d'interpréter un panel étendu d'interactions humaines mais également de restituer une réponse intuitive à l'interlocuteur en utilisant plusieurs modes.

2.2 Objectifs

Dans le contexte cité précédemment il apparaît pertinent de munir un robot, véritable agrégateur de capteurs et d'actionneurs, d'un moteur de dialogue multimodal. Ce projet est le cœur de mon sujet de stage de fin d'études. Bien qu'en constante évolution au cours de mes travaux l'application initiale du moteur de dialogue tourne autour de l'accueil et l'accompagnement de personnes, qu'elles soient des employés d'Orange, des visiteurs sur site ou encore des clients en boutique. La visée du projet se concentre donc essentiellement sur la robotique à but social.

L'objectif principal d'un tel système est de doter un robot des capacités tacites d'un humain à communiquer. En effet bien qu'un dialogue soit souvent assimilé à un échange verbal de mots, d'innombrables informations supplémentaires sont transmises. Bien interprétées ces dernières permettent l'identification précise de l'état émotionnel et de l'intention de l'interlocuteur, il est alors possible d'adapter la réaction du robot en conséquence, ce que ferait instinctivement un humain.

2.3 Intérêts

Outre le défi technique, le mise en œuvre d'un dialogue multimodal présente de nombreux intérêts. En premier lieu l'interlocuteur naïf gagne un temps considérable à interagir avec un système qui ne suppose aucune connaissance *a priori* et qui, de plus, comprend un hochement un tête là où la faible fiabilité des systèmes de reconnaissance vocale l'aurait forcé à répéter son ordre à plusieurs reprises, dans le contexte d'un environnement bruyant par exemple. Le dialogue multimodal permet dans le pire des cas une compréhension au niveau du dialogue unimodal. Il peut saisir des ordres complexes et la redondance des informations lui offre souvent un moyen de conforter ses interprétations. Cependant, la principale difficulté dans la mise en œuvre de ces systèmes est le traitement des cas où les différents modes d'interactions fournissent des informations contradictoires.

De plus, bien que la vocation principale du projet robotique soit la veille technologique, c'est à dire l'occasion pour Orange de garder un pied dans la robotique, l'entreprise peut tirer bénéfice de ces développements. En effet un tel moteur de dialogue ne se limite pas à une plateforme robotisée mais peut être utilisé sur tout appareil possédant des capteurs et/ou des actionneurs. Des champs d'applications au cœur des métiers d'Orange sont par exemple la maison intelligente ou les enceintes connectées. Finalement la force d'Orange est sa pluridisciplinarité, de nombreux outils développés en interne sont disponibles librement pour les projets de l'entreprise et les travaux de plusieurs équipes viennent étoffer les technologies mises à disposition. Cela permet une meilleure maîtrise des outils utilisés et une indépendance totale des fournisseurs tiers.

2.4 Difficultés

Bien que possédant maints avantages, les difficultés d'implémentations d'un système multimodal sont nombreuses et il est en pratique difficile de mettre en œuvre une telle plateforme. D'une part la multimodalité se base essentiellement sur l'interprétation de données, il est malheureusement bien souvent difficile d'estimer la fiabilité de ces interprétations, d'autant que nombres de ces dernières sont calculées et traitées par des algorithmes de reconnaissance très sujets aux erreurs.

D'autre part, il est essentiel qu'un système de dialogue soit capable de traiter les informations en temps réel afin d'offrir une interaction fluide à l'interlocuteur. La fluidité est en effet une composante fondamentale car l'utilisateur doit pouvoir deviner intuitivement ce que fait le robot à tout moment, comme il le ferait avec un humain. En cas de traitement long, l'attente du résultat d'une requête HTTP par exemple, il est primordial d'informer l'interlocuteur qu'il va devoir patienter quelques instants, que ce soit avec un mouvement de bras, un clignotement de LEDs ou encore avec un message vocal. Des réponses rapides et un retour permanent de l'état interne du robot sont finalement des contraintes essentielles pour pouvoir mettre en place une interaction ergonomique et intuitive.

De plus, considérer un nombre important de modes de communication peut entraîner une multitude de cas d'ambiguïté. Cette problématique peut être abordée de plusieurs manières différentes, en priorisant certains modes par rapport à d'autres par exemple, ou encore en considérant la première interprétation obtenue comme celle de référence. Par simplicité, la solution retenue dans les travaux de mon stage est de demander une désambiguïsation à l'utilisateur dès lors que des informations contradictoires sont interprétées.

Finalement la dernière difficulté majeure est la fusion des données issues des capteurs. L'enjeu est de trouver le niveau de fusion des données qui permet une interprétation fiable tout en restant simple et modulaire. A titre d'exemple on peut étudier la modalité *voix*. Dans le cas d'une fusion bas niveau on utilise le signal audio brut du microphone qui possède le plus d'informations mais peu explicites, en progressant d'un niveau on utilise la chaîne de caractères correspondant à la phrase prononcée par l'interlocuteur. Enfin pour une fusion haut niveau on utilise l'analyse sémantique de la phrase qui fournit les résultats les plus exploitables mais possède assez peu d'informations en comparaison du signal brut. Il est donc nécessaire de trouver le bon compromis entre exhaustivité et pertinence.

2.5 Cadre de l'étude

Le sujet proposé possède de nombreuses facettes et difficultés. Afin de simplifier ce dernier on considère dans tous les cas d'usage mis en œuvre que le robot a systématiquement un unique interlocuteur, ainsi les modalités d'entrée ont toutes la même origine. De plus on considère qu'un interlocuteur parlant à proximité du robot s'adresse directement à lui.

Pour ce qui est de l'environnement de développement, le cœur du système fonctionne avec le middleware open-source *Robot Operating System* (ROS), très utilisé en robotique pour sa modularité et sa flexibilité. Il s'impose aujourd'hui comme un standard de la recherche en robotique et est utilisé dans certains robots industriels comme le PR2 de Willow Garage ou le Robonaut 2 de la NASA actuellement à bord de la station spatiale internationale. La version de ROS utilisée pour le projet est *Kinetic Kame* sous Ubuntu 16.04, choix motivé par la longévité de son support officiel.

Son principe de fonctionnement repose sur un système multi-processus appelés *nœuds*. Ces derniers peuvent s'interfacer à l'aide de canaux bi-directionnels TCP appelés *topics*. Du fait de l'utilisation de la couche réseau pour la communication entre les nœuds il est d'une part possible d'interfacer des processus développés dans des langages différents. Dans notre cas cela permet une grande flexibilité de développement, les nœuds critiques étant développés en C++ et les autres en Python. Les modules Android développés en Java deviennent également compatibles avec l'ensemble. D'autre part le réseau offre la possibilité de déporter les calculs sur plusieurs ordinateurs de manière transparente ce qui s'avère pertinent au vue des ressources limitées embarquées par le robot. Ce système est régi par un ordinateur maître appelé *rosmaster* agissant comme référence pour les ins-

tanciations des transactions TCP.

Le middleware ROS tire sa force de sa communauté très active et de son côté open-source. En effet bien qu'imaginé et maintenu par l'entreprise Willow Garage une partie des paquets ROS sont le résultat du travail de la communauté, qu'ils soient de simples passionnés ou des chercheurs. Il y a également une forte diversité dans les pilotes de capteurs et actionneurs disponibles pour la plateforme, ce qui rend leur intégration particulièrement aisée.

Chapitre 3

État de l'Art

3.1 Multimodalité

Étudiée pour la première fois par Richard A. Bolt en 1980[2], la multimodalité alors limitée par les puissances de calculs n'est réellement considérée qu'au début des années 2000. La majorité des travaux de recherche de cette période se focalisent sur la multimodalité en tant qu'alternative aux interfaces homme-machine habituelles composées d'un écran et de périphériques d'entrée comme la souris ou le clavier. Aujourd'hui, de nouveaux périphériques voient le jour avec des capteurs permettant une caractérisation de leur environnement précise et variée. L'emploi des techniques d'apprentissage automatique rend possible l'interprétation de comportements humains complexes comme la gestuelle ou l'état émotionnel. Ces avancées offrent la possibilité de concevoir des systèmes de dialogue avancés prenant en considération l'ensemble des modes d'interaction plutôt que d'opérer une fusion à haut niveau de compréhension souvent imprécise.

Outre les systèmes d'information usuels la multimodalité est particulièrement adaptée à la robotique, en effet de par ses capteurs et actionneurs un robot peut être doté de fonctionnalités propres à l'humain comme la préhension, la parole ou encore le mouvement. Parfois ces robots sont conçus pour ressembler à un être humain ou imiter en partie ses caractéristiques. Font partis de cette catégorie le robot Waldo utilisé lors de ce stage ainsi que le très connu robot Pepper de SoftBank Robotics. Ce dernier, robot humanoïde à l'apparence enfantine, a été le support de nombreux travaux sur la multimodalité[10] et apparaît comme un choix pertinent lorsqu'il s'agit d'interagir avec des utilisateurs naïfs peu habitués au contact avec la technologie comme les enfants ou les personnes âgées[6]. Entre autres, ce contact avec un robot participe grandement au bien-être des personnes malades ou souffrant d'handicaps particuliers, établir un dialogue naturel et rassurant est donc aussi un enjeu de la multimodalité.

D'autres robots n'ont pas vocation à ressembler à l'humain, c'est le cas notamment du robot Spoon ci-contre qui s'inspire très largement des comportements animaux. En plaçant l'émotion au cœur de son système, Jérome Monceaux, le concepteur de Spoon, exploite la multimodalité afin de créer un robot doté d'une personnalité, capable de réagir au gré de son environnement et d'évoluer[7]. Avec cet exemple il n'est finalement pas juste d'associer dialogue multimodal et robot humanoïde, la multimodalité n'étant pas propre à l'humain mais bien aux interactions en général.

FIGURE 3.1 – Jérome Monceaux et son robot Spoon

Bien qu'il est faux de considérer qu'un humain interagit avec un robot de la même façon qu'avec l'un de ses semblables[9], il est tout à fait pertinent de le doter de capacités d'interaction humaines. Un utilisateur ne va pas nécessairement parler à un robot ou tenter un contact physique avec lui, cependant une réponse adéquate de ce dernier peut installer un climat propice au dialogue. En effet l'humain se base sur les réactions de son interlocuteur pour la poursuite du dialogue, il est donc primordial de ne pas négliger les modalités de sortie qui forment le cœur du comportement du robot afin de proposer un échange cohérent avec ses habitudes de communication.

3.2 Interactions Multimodales

Afin d'appréhender la multimodalité il est important de distinguer les différents types d'interactions multimodales. Cependant les particularités des ces interactions empêchent l'utilisation des modèles conventionnellement utilisés pour les interactions pensées avec le paradigme WIMP [1].

Le modèle CARE, proposé par Joëlle Coutaz en 1995[3], est encore aujourd'hui une référence dans ce domaine. Ce modèle classe les interactions multimodales en quatre catégories différentes qui sont respectivement *Complementarity* (Complémentarité), *Assignment* (Assignation), *Redundancy* (Redondance) et *Equivalence* (Équivalence). Ces différentes classes sont explicitées dans le tableau 3.1 ci-contre.

Dans le cas des systèmes de dialogues unimodaux habituels seules les interactions d'assignation et de redondance sont considérées, bien que la redondance ne soit que très rarement exploitée. La multimodalité permet quant à elle de considérer l'ensemble de ces

1. WIMP : Windows, Icons, Menus and Pointing device. Paradigme des bases fonctionnelles d'une interface graphique en informatique.

Complémentarité	Plusieurs modalités séquentielles ou combinées pour une action
Assignation	Une unique modalité pour une action
Redondance	Plusieurs modalités simultanées pour une même action
Équivalence	Plusieurs modalités pour une même action, une seule suffit

Table 3.1 – Modèle d'interactions multimodales CARE

interactions, aussi bien en entrée qu'en sortie. Néanmoins dans le cadre de mon stage il ne sera pas fait mention des interactions complémentaires dont l'interprétation nécessite plus qu'un stage de fin d'études.

3.3 Langages de description

Tout dialogue se doit d'être configuré en accord avec le cas d'usage à implémenter, de nombreux langages de description existent à cet effet. L'organisation W3C [2] a créé EMMA[8], un standard sous la forme d'un langage de balisage destiné à représenter les données d'interactions multimodales. Ce langage est très polyvalent et permet de considérer de nombreux *inputs* différents cependant il a été conçu pour le web multimodal et est dans ce sens adapté à l'interprétation par un navigateur. De plus c'est un langage orienté machine avec une description au niveau des entrées et sorties, aux antipodes d'une description événementielle ou en encore de dialogue.

Un autre langage particulièrement intéressant est le SMUIML[4] [3], conçu pour le multimodalité il offre la possibilité de définir les différentes modalités d'entrée, de sortie, ainsi que la logique régissant le dialogue. Pensé pour être humainement compréhensible, il utilise le langage de balisage XML [4]. Il souffre cependant d'un manque d'adaptabilité au contexte ainsi qu'aux utilisateurs, en outre il ne prévoit pas un traitement des cas d'erreurs.

Il existe de nombreux autres langages de description pour les dialogues multimodaux, il est cependant difficile de trouver un compromis entre facilité de lecture et polyvalence. Il ressort de la littérature que chaque langage a ses propres forces et faiblesses, il convient donc d'adapter son choix en fonction de l'application.

2. World Wide Web Consortium
3. SMUIML : Synchronized Multimodal User Interaction Modeling Language
4. XML : Extensible Markup Language

3.4 Fusion

L'une des différences fondamentales entre les différents systèmes multimodaux réside dans le niveau de fusion des données auquel ils opèrent. En effet l'humain extrait naturellement de robustes informations à partir des signaux captés par l'ensemble de ses sens, le défi ici est de concevoir un système capable de se comporter de manière similaire. Cependant le niveau de fusion de ces informations est crucial pour leur interprétation, il s'agit en effet d'évaluer l'ensemble plutôt que de considérer chaque modalité comme indépendante. L'idéal serait de considérer tous les niveaux en même temps mais il est difficile de concevoir et implémenter un tel système informatique en pratique.

Afin de bien cerner la problématique il est important de distinguer les différents niveaux de fusion qu'il est possible d'exploiter. La littérature[12][1] en dénombre généralement cinq :

- **Fusion des données brutes :** Peu utilisée en pratique à juste titre, il est difficile de fusionner une tableau de pixels provenant d'une image avec un texte issu d'un algorithme de reconnaissance vocale.
- **Fusion des caractéristiques :** Souvent décrite comme la fusion prématurée, cette stratégie consiste à concaténer les données synchronisées de chaque capteur afin de classifier l'ensemble.
- **Fusion des classificateurs :** Stratégie de fusion intermédiaire, il s'agit ici de classifier les données issues des capteurs puis fusionner les classificateurs obtenues.
- **Fusion au niveau décisionnel :** Fusion tardive, cette stratégie implique l'utilisation de scores de confiance pour chaque interprétation, souvent calculés par des classificateurs indépendants. Il est alors possible d'interpréter l'ensemble avec un système de poids associés à chaque modalité et chaque interprétation.
- **Fusion au niveau sémantique :** Chaque modalité est traitée indépendamment et analysée sémantiquement puis fusionnée avec les autres. C'est le niveau de fusion le plus élevé mais aussi le moins précis, on considère ici chaque source comme indépendante entraînant ainsi une importante perte d'information.

Chapitre 4

Moteur de Dialogue Multimodal

4.1 Environnement de Développement

La plateforme de développement utilisée pour le projet est le robot Waldo présenté dans la partie 1.2.3. La diversité des capteurs présents sur ce dernier offrent la possibilité de considérer plusieurs modes d'entrée et de sortie :

- **Entrée**
  - Microphone : Reconnaissance vocale, bruit ambiant
  - Caméra : Reconnaissance de gestes, de visages, d'objets
  - Caméra 3D : Détection d'environnement, localisation spatiale d'objets
  - Tablette : Interaction tactile
  - LIDAR : Détection d'environnement ou de présence humaine
- **Sortie**
  - Haut-Parleurs : Synthèse vocale, sons
  - LEDs : Indication d'état
  - Tablette : Affichage visuel
  - Bras : Préhension, gestes

Comme détaillé précédemment dans la partie 2.5, le support de développement de l'équipe robotique est le middleware ROS. Ces particularités en font un choix pertinent pour un système de dialogue multimodal. En effet en considérant les différentes modalités -en entrée et sortie- comme des modules indépendants il devient facile d'ajouter ou supprimer ces modules. De plus sa flexibilité est particulièrement adapté à notre plateforme multi-capteurs et multi-actionneurs. Les modules du dialogue multimodal sont développés en Python par soucis de rapidité, de plus ils ne nécessitent pas de ressources matérielles importantes.

4.2 Mise en Œuvre du Dialogue

4.2.1 Langage de Description

Parmi les langages de description existants mentionnés dans la partie 3.3 de mon état de l'art, il est difficile de trouver un langage à la fois polyvalent et évolutif. J'ai finalement choisi de créer ma propre convention en m'inspirant très largement du langage SMUIML[4] qui m'a paru être le choix le plus pertinent. Tout d'abord il est assez rapide à mettre en place pour un dialogue simple, il est de plus très modulaire et permet une évolution aisée.

Le principe majeur est de séparer la configuration d'un dialogue en trois parties avec en premier lieu les modalités d'entrée et leur fusion, ensuite la logique de déroulement du dialogue et enfin les modalités de sortie. Ces trois configurations sont traitées par trois composants logiciels, respectivement le moteur de fusion, le moteur de dialogue et le moteur de fission, le tout formant le moteur multimodal. Chaque partie se configure avec un fichier à part entière dont le formalisme est le JSON [1], ainsi trois fichiers suffisent à configurer un dialogue multimodal complexe dans son intégralité.

4.2.2 Logique de Dialogue

La multimodalité étant orchestrée par les moteurs de fusion et de fission, la logique de dialogue est quant à elle gérée par le moteur de dialogue. Ce dernier se base sur une machine à états finis dont les transitions sont les actions déclenchées par les interactions de l'utilisateur. Ce choix d'implémentation a été motivé d'une part par sa robustesse, et d'autre part par sa flexibilité d'implémentation. Une machine à états finis est en effet robuste car déterministe, chaque état et transition est défini lors de la configuration. Avec un tel système il devient facile d'estimer, à partir d'un état donné, les suites possibles du dialogue et ainsi limiter les sources aux nécessaires à chaque instant.

4.2.3 Exemple

Cas d'Usage

L'exemple détaillé ici se concentre sur la configuration d'un dialogue multimodal simple avec un robot. Les moyens de captation du robot sont un microphone, l'écran tactile d'une tablette et une caméra pour la reconnaissance de gestes. Il est également muni d'actionneurs et de composants communicants, respectivement des haut-parleurs et une tablette.

Il s'agit dans un premier temps d'initier l'interaction, on laisse ici le choix à l'utilisateur de saluer de la main le robot ou d'opérer un clic sur la tablette. Le robot peut aussi prendre l'initiative du dialogue en engageant la conversation quand un interlocuteur est

1. JSON : JavaScript Object Notation

présent devant lui depuis un certain temps. Ces trois interactions sont autant de moyens valides d'initier le dialogue. Après avoir salué son interlocuteur le robot lui demande si il souhaite voir une personne, lui laissant alors la possibilité de détailler son intention par une réponse orale ou en tapant le nom d'une personne à l'aide de la tablette tactile. Par soucis de simplicité nous supposons ici que le nom renseigné correspond systématiquement à une personne connue et à proximité. Le dialogue prend alors fin et le nom communiqué est envoyé au module de navigation afin de guider l'utilisateur vers sa destination.

Configuration des Entrées

Il s'agit tout d'abord de définir les moyens avec lesquels l'utilisateur pourra communiquer avec le robot, les modalités d'entrée. Dans notre cas d'usage le robot met à profit trois capteurs, un microphone, l'écran d'une tablette tactile et une caméra pour la reconnaissance de gestes respectivement associés aux modalités *voice*, *tablet* et *gesture*. Toute la configuration des entrées s'effectue dans le fichier JSON *input.json* en respectant le formalisme détaillé dans l'annexe A.

En premier lieu est effectuée la définition des modalités d'entrée comme présenté dans le code source 1, c'est à dire des sources acceptées par le moteur multimodal. Il est également possible d'indiquer les modalités pouvant mettre fin à une interaction venant du robot à l'aide du paramètre *bargein*. Lorsque ce dernier est activé une interaction utilisateur utilisant la modalité concernée peut *couper la parole* au robot, c'est à dire interrompre le robot pendant son processus d'interaction. Dans notre cas d'usage un clic sur la tablette ou un geste est susceptible d'interrompre le robot, ce qui n'est pas le cas de la voix qui devient difficile à interpréter lorsque le robot émet lui même des sons.

```
"modalitites": [
  {
    "name": "voice",
    "bargein": false
  },
  {
    "name": "tablet",
    "bargein": true
  },
  {
    "name": "gesture",
    "bargein": true
  }
]
```

Code Source 1 – Exemple de configuration des modalités d'entrée dans le fichier *input.json*

La configuration se poursuit avec les déclencheurs, ou *triggers*, qui définissent des valeurs clés pour chaque modalité. Ainsi lorsqu'une interaction est interprétée par le biais d'une modalité sa valeur et son mode sont comparés aux déclencheurs, si il y a une correspondance le trigger est déclenché. Pour le cas d'usage il est nécessaire de définir notamment les déclencheurs de la première interaction, le code source 2 montre deux d'entre eux, respectivement *gesture_hello*, correspondant à un salut de la main capté par la modalité *gesture*, et *voice_hello*, déclenché lorsque la modalité *voice* capte le mot "*bonjour*".

```
"triggers": [
  {
    "name": "gesture_hello",
    "source": "gesture",
    "value": "hello"
  },
  {
    "name": "voice_hello",
    "source": "voice",
    "value": "bonjour"
  },
  { "...": "..." }
]
```

Code Source 2 – Exemple de configuration des déclencheurs dans le fichier *input.json*

Finalement il faut définir les actions, cœur des interactions multimodales. Une action se caractérise par un ensemble de déclencheurs conditionnant son exécution et une fenêtre temporelle. Si une combinaison adéquate de déclencheurs survient dans un laps de temps inférieur à la fenêtre temporelle, l'action est alors exécutée. A titre d'exemple le code source 3 montre la définition de l'action initiatrice du dialogue de notre cas d'usage. Celle-ci est exécutée à la suite du déclenchement du trigger *gesture_hello*, *voice_hello* ou les deux si ils sont captés dans un intervalle de temps inférieur à 0.5 secondes.

Un tel fonctionnement permet en outre de considérer les interactions multimodales complémentaires ignorées jusqu'ici. En effet en ne gardant que la troisième ligne des *triggers* de l'action issue de l'exemple le dialogue va imposer la présence des deux déclencheurs.

```
"actions": [
  {
    "name": "hello",
    "timeWindow": 0.5,
    "triggers": [
      "gesture_hello",
      "voice_hello",
      "voice_hello && gesture_hello"
    ]
  },
  { "...": "..." }
]
```

Code Source 3 – Exemple de configuration des actions dans le fichier *input.json*

Configuration du Dialogue

La deuxième étape consiste à construire la machine à états finis régissant la logique du dialogue, ses états et ses transitions. Les états correspondent aux attentes d'interaction du robot et les transitions aux actions définies dans le fichier précédent. Dans l'exemple du code source 4 deux états sont configurés ainsi que la transition les liant, transition déclenchée par l'action *hello* configurée précédemment. On suppose ici que l'état *start* est l'état dans lequel se trouve le dialogue initialement, cet exemple montre alors la première progression de la logique de dialogue lors de l'initiation de ce dernier. La configuration du dialogue s'effectue dans le fichier JSON *dialogue.json* en respectant le formalisme détaillé dans l'annexe B.

```
"states": [
  {
    "name": "start"
  },
  {
    "name": "introduction"
  },
  { "...": "..." }
],

"transitions": [
  {
    "source": "start",
    "trigger": "hello",
    "dest": "introduction"
  },
  { "...": "..." }
]
```

Code Source 4 – Exemple de configuration du fichier *dialogue.json*

Configuration des Sorties

Finalement ne reste plus qu'à configurer les moyens d'expression du robot, les modalités de sortie. De manière similaire à celles d'entrée, il faut tout d'abord définir chaque mode de communication comme détaillé dans le code source 5. Le champ *topic* correspond au nom du canal de communication ROS que le moteur de fission doit utiliser pour assurer l'exécution de l'expression. Les modules de sortie écoutent donc sur leur topic, réalisent l'interaction lorsque sollicités et informent par la suite le moteur de fission de la fin du processus. La configuration des modalités de sortie s'effectue dans le fichier JSON *output.json* en respectant le formalisme détaillé dans l'annexe C.

```
"modalities": [
  {
    "name": "voice",
    "topic": "tts"
  },
  {
    "name": "tablet",
    "topic": "tablet"
  },
  {
    "name": "sound",
    "topic": "sound"
  },
  { "...": "..." }
]
```

Code Source 5 – Exemple de configuration des modalités de sortie fichier *output.json*

Enfin il faut spécifier les informations à envoyer aux différents modules d'interaction, un exemple est visible sur le code source 6. Chaque transition effectuée par le moteur de dialogue peut coïncider avec une expression, le champ *action* doit pour cela correspondre à l'action déclenchée en entrée. Par la suite il est possible de définir les données allant être envoyées aux modalités définies ci-dessus. Ici les modules *voice* et *tablet* vont recevoir la même chaîne de caractères, l'un va l'afficher et l'autre la prononcer, enfin la modalité *sound* reçoit le nom du fichier audio qu'elle doit jouer.

```
"outputs": [
  {
    "action": "hello",
    "voice&&tablet": "Bonjour, comment puis-je vous aider ?",
    "sound": "Hello.wav"
  },
  { "...": "..." }
]
```

Code Source 6 – Exemple de configuration des sorties du fichier *output.json*

Conclusion

A travers cet exemple minimaliste et non-exhaustif on peut constater que la configuration d'un dialogue est certes longue mais simple et le tout est maintenable et évolutif selon les besoins.

4.3 Moteur de Dialogue

4.3.1 Principe

La modularité de ROS évoquée précédemment a fortement dirigé mes choix d'implémentation vers une architecture modulaire. Bien que le cas droit du projet soit pourvu de modalités d'entrée et de sortie bien définies, il paraissait pertinent de créer un système pouvant s'adapter simplement à des changements de configuration et/ou de matériel. ROS a finalement très fortement influencé l'architecture macroscopique du système.

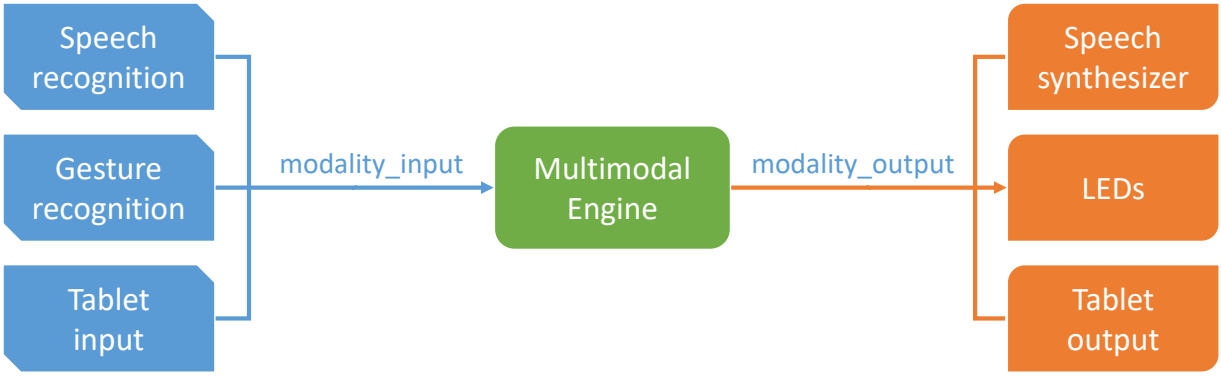

Figure 4.1 – Fonctionnement macroscopique du moteur multimodal

Le fonctionnement général du système est présenté sur la figure 4.1. Le moteur de dialogue multimodal écoute en premier lieu les interactions en entrée par le biais d'un topic ROS dont les messages sont clairement définis. Ces messages doivent posséder notamment une date qui servira de temps de référence pour l'interaction, une chaîne de caractères correspondant à l'une des modalités d'entrée définies dans les fichiers de configuration et une chaîne de caractères traduisant l'interprétation de l'interaction de l'interlocuteur. Le fait de contraindre chaque module d'entrée à réduire son interprétation à une chaîne de caractères n'est pas anodin, cela permet une standardisation des modules, c'est de plus la conséquence d'une fusion à haut niveau. Pour chaque mode il est finalement nécessaire de mettre en place le contrat d'interface avec le moteur multimodal.

Pour ce qui est des modalités de sortie le principe est proche, chaque module doit implémenter un *actionlib server* [2] avec le contrat d'interface imposé par le moteur multimodal. Lorsqu'il est nécessaire ce dernier envoie les ordres adaptés à la situation aux différents modules et attend que chacun termine sa tâche. A titre d'exemple si le module de sortie est un module de synthèse vocale, le moteur multimodal va attendre la fin de la phrase avant de poursuivre les routines de la logique de dialogue.

2. Actionlib Server : Topic ROS bidirectionnel destiné aux tâches longues

4.3.2 Difficultés

La difficulté de conception dans ce type de système est de permettre une grande flexibilité et modularité tout en restant facilement configurable. La plupart des échanges de parole nécessitent ponctuellement l'appel de fonctions pour effectuer des tâches complexes qui ne sont pas du ressort du dialogue. Par exemple effectuer la recherche d'un nom dans un annuaire n'est pas inné pour le système de dialogue mais nécessaire pour la poursuite de l'échange. La solution mise en œuvre ici est la possibilité à chaque transition de la machine à états finis d'appeler une routine Python afin d'effectuer des tâches annexes au dialogue.

Une autre difficulté réside dans la synchronicité des interactions, considérer les fenêtres de temps définies pour les actions et déterminer en temps réel leur exécution. Afin de ne pas considérer trop hâtivement une action au détriment d'une autre, les modules des modalités d'entrée ont la possibilité d'informer le moteur de dialogue qu'une interaction se produit sans pour autant y joindre des données relatives à l'interprétation. Ce fonctionnement est particulièrement utile pour les interactions longues, nécessitant des calculs lourds ou des requêtes web. C'est le cas notamment de la reconnaissance vocale, il est en effet facile de connaître le temps exact où l'interlocuteur commence à parler mais l'attente d'une interprétation sous forme de texte peut parfois prendre du temps, d'autant plus que dans notre cas la reconnaissance est effectuée par un serveur distant. Le moteur de dialogue doit donc être informé qu'il doit attendre une interprétation de la part du module avant de confirmer l'exécution d'une action.

4.3.3 Critique

Finalement le moteur de dialogue multimodal mis en œuvre ici répond aux attentes et implémente bien la multimodalité comme escompté. On peut noter en outre qu'il est aisément configurable mais suppose tout de même un niveau minimum en informatique, un développement futur pourrait par exemple y ajouter une interface graphique.

La limitation majeure de cette implémentation réside selon moi dans le déterminisme imposé par la machine à états finis régissant la logique du dialogue. Je pense qu'il serait pertinent de modifier ce système en le remplaçant par des chaînes de Markov cachées[5] par exemple, ou un réseau de neurones[10]. Ces deux méthodes ont l'avantage indéniable d'être auto-apprenantes, ainsi la logique de dialogue évolue au fil des expériences personnelles du robot.

Chapitre 5

Organisation et Activités du Projet

5.1 Maintenance

5.1.1 Maintenance Logicielle

Afin de mutualiser nos travaux nous avons mis en place un dépôt Git, gestionnaire de version devenu un standard dans l'informatique aujourd'hui. Cette initiative a non seulement permis de développer une plus grande rigueur dans les développements mais elle a aussi accru la cohésion et l'organisation de l'équipe. En effet lorsqu'une équipe travaille dans un même dépôt il devient nécessaire de converser régulièrement afin de se mettre d'accord sur des processus d'intégration définis, cette mise en place a fortement participé à l'application de la méthodologie agile au sein du projet, comme détaillé ci-après dans la partie 5.2.2.

Les codes disponibles sur Waldo et fournis par Immersive Robotics étant en constante évolution il s'est souvent avéré nécessaire de procéder à des refontes en profondeur du système. Cela nous a notamment permis de mieux comprendre le fonctionnement du robot ainsi que les dépendances de chaque module. En outre il est intéressant de constater une utilisation professionnelle du middleware ROS.

5.1.2 Maintenance Matérielle

Similairement à la partie logicielle, les composants matériels de Waldo ainsi que son câblage sont en constante évolution. Le faible volume du corps du robot est par exemple un facteur primordial à prendre en compte lorsqu'il s'agit d'ajouter ou de déplacer un composant. De plus nous avons été confronté à plusieurs reprises à un problème de bande passante USB limitée nous obligeant à repenser le câblage dans sa globalité. La solution adoptée a finalement été d'ajouter un Odroid, micro-ordinateur équivalent à un Raspberry Pi, sur lequel sont déportés les périphériques USB haut-débit tels que les caméras.

5.2 Robot Chercheur-Trouveur

5.2.1 Scénario

Chaque année est organisé le Salon de la Recherche des Orange Labs (SROL) où des visiteurs aussi bien internes qu'externes peuvent avoir un aperçu des différents travaux actuels de l'entreprise. Chaque équipe projet est donc invitée à proposer un démonstrateur si leurs travaux se prêtent à l'exercice et que la maturité du projet est suffisante. Suite aux réunions de l'équipe robotique nous avons décidé de proposer un robot chercheur-trouveur d'objets pour une démonstration au salon de la recherche. Bien que ce démonstrateur n'ai pas été retenu pour le salon nous avons poursuivi son développement.

Baptisé *Seeky Finder*, le service du robot chercheur-trouveur a pour objectif de répertorier les différents objets d'un environnement afin de faciliter leur recherche ou effectuer un inventaire. Un cas d'usage est par exemple la réalisation d'un inventaire de magasin ou d'entrepôt, le robot est alors capable de gérer les stocks ou encore guider une personne vers un produit particulier. Un second cas d'usage est l'assistance des personnes âgées ou à mobilité réduite, le service permettant de les accompagner dans leur recherche des objets du quotidien.

La mise en place de ce scénario nécessite une importante diversité de modules logiciel techniques. Le cœur du service repose sur la capacité du robot à se déplacer et se repérer dans un environnement connu d'une part, et à détecter des objets à partir d'images prises par ses caméras d'autre part. Viennent ensuite s'ajouter en périphérie des besoins fonctionnels comme la détection d'humains permettant au robot de prendre l'initiative du dialogue lorsqu'un interlocuteur s'approche de lui. Il est également nécessaire de pouvoir mener un dialogue avec ce dernier afin de cibler son besoin. Enfin, le bras robotisé dont Waldo dispose peut être utilisé comme dispositif de pointage en cas de réussite de la recherche. Ce service met finalement à profit l'intégralité des travaux de chacun des membres de l'équipe.

5.2.2 Méthodologie Agile

Nos travaux n'étant plus indépendants nous avons décidé de régir notre travail d'équipe selon une méthode agile, pratique particulièrement valorisée par Orange aujourd'hui. Très répandues dans les grandes entreprises et particulièrement dans le secteur de l'informatique, les méthodes agiles sont nombreuses et il est nécessaire d'adapter son choix en fonction des besoins du projet afin d'en tirer une efficacité maximum. Étant peu expérimentés dans ce domaine nous avons sollicité l'aide d'une formatrice interne pour nous aiguiller.

La méthode agile finalement retenu est le Kanban, elle permet de bien gérer le flux des tâches et offre une très bonne visibilité sur le long terme. Son avantage majeur dans pour nous est la possibilité d'ajouter des tâches au besoin sans surcharger le reste de la chaîne

de développement ce qui n'est pas le cas du Scrum par exemple. Le principe fondamental du Kanban est tout d'abord le découpage du travail en tâches, ces dernières vont ensuite transiter à travers plusieurs étapes de manière séquentielle en commençant par l'étude, puis la conception et enfin la validation. Des étapes intermédiaires peuvent être intégrées, il est nécessaire de les décider en fonction des rôles de chacun et du type de projet. Dans notre cas nous avons mis en place quatre étapes :

- Étude et conception
- Développement
- Démonstration
- Intégration et tests fonctionnels

A chaque étape correspond des critères devant être validés afin de pouvoir la considérer comme terminée. A titre d'exemple l'étape d'étude est considérée comme terminée lorsque les technologies mises en œuvre ont été identifiées, que chacun a les compétences nécessaires pour réaliser la tâche et qu'une ébauche de documentation est écrite. Chaque tâche va donc transiter à travers la chaîne de développement jusqu'à réalisation du scénario.

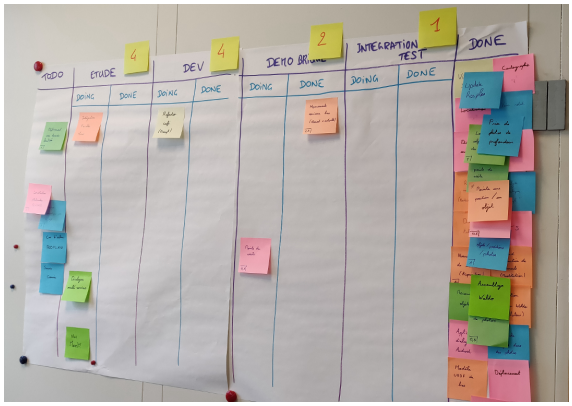

FIGURE 5.1 – Tableau agile utilisé par l'équipe robotique

Une telle méthodologie offre une bonne visibilité sur les travaux en cours et permet de mieux estimer le temps nécessaire à la mise en place d'un projet. De plus nous avons constaté une meilleure rigueur et une efficacité accrue au fil de l'avancée du projet. La communication dans l'équipe s'est également améliorée ce qui a permis une compréhension plus complète du projet.

5.2.3 Détection d'Objets

Tensorflow

Afin de mettre en place le scénario décrit précédemment, il est nécessaire de développer un module capable de détecter des objets à partir d'images. Pour cela il est nécessaire d'avoir recours à des modèles empiriques issus de l'apprentissage automatique tant les représentations visuelles de plusieurs objets de la même catégorie sont diverses et tant les caractéristiques des images -contraste, luminosité, résolution- sont variables.

Une solution pragmatique et efficace à ce problème est le très connu framework d'apprentissage automatique Tensorflow développé par Google. La force de cet outil est qu'il

propose une importante palette de modèles pré-entraînés sur des banques d'images contenant plusieurs dizaines d'objets de la vie courante. Les phases d'élaboration d'un corpus d'images et d'entraînement étant les plus coûteuses en temps et en ressources, il devient alors immédiat de mettre en place une chaîne de traitement destinée à la détection d'objets dans les images.

Responsable du développement de ce module, j'ai eu le choix entre deux alternatives d'implémentation. D'une part j'ai considéré l'utilisation de Tensorflow sur un ordinateur local, et d'autre part j'ai envisagé l'utilisation d'une API interne à Orange proposant des services de traitement d'images.

Implémentation Locale

Tout d'abord, afin de déterminer le modèle le plus adapté à une implémentation locale, j'ai réalisé de nombreux tests sur des images prises en conditions réelles, soit avec les caméras du robot et dans un environnement de test adapté aux besoins du scénario. Deux critères principaux ont été décisifs pour le choix du modèle, la capacité de détection des petits objets du quotidien et le temps de calcul pour une image standard [1]. En effet dans notre cas il ne s'agit pas de détecter un avion, un bus ou encore un canapé mais plutôt des objets de la taille d'une tasse ou d'un téléphone. De plus les ressources matérielles mises à disposition [2] ne permettent pas d'égaler les mesures présentées sur la notice GitHub des modèles de Tensorflow[11].

Après considération de l'ensemble, le modèle le plus adapté à notre cas d'usage est le modèle *faster_rcnn_inception_v2_coco* entraîné sur la banque d'images COCO [3]. Ce dernier est capable de détecter plus de 90 objets différents et le traitement moyen d'une image est d'environ 5,8 secondes ce qui est satisfaisant bien que loin du temps réel. De plus l'écart-type du temps de traitement étant très faible il est aisé d'estimer les temps de traitement globaux du scénario.

Utilisation de l'API d'Orange

La deuxième solution d'implémentation est l'utilisation d'une API interne à Orange dédiée au traitement d'images. Elle propose des services comme la détection et la reconnaissance de visages, la détection des émotions ou encore la détection d'objets. Bien que la majorité de ces services sont basés sur des technologies internes, la détection d'objets est quant à elle effectuée à l'aide de Tensorflow ce qui permet une comparaison équitable avec la solution précédente.

1. Ici une image compressée au format JPEG de résolution 640x480
2. Une carte graphique Nvidia GeForce GTX 1050
3. COCO : Common Objects in Context

En utilisant l'API d'Orange la détection se fait en deux étapes. Tout d'abord est effectuée une première requête HTTP [4] afin de téléverser l'image sur les serveurs distants, puis une deuxième requête HTTP afin de demander la détection d'objets sur l'image récemment envoyée. Du fait de l'instabilité du réseau il est difficile d'estimer un temps de traitement pour une image, cependant durant nos tests l'ensemble de la chaîne -téléversement et détection d'objets- a rarement dépassé les 5 secondes. De plus, la réalisation du processus ne consommant que très peu de ressources matérielles, il est possible de faire appel au multithreading pour paralléliser les analyses. Il est toutefois important de noter que des quotas étant en place, demander l'analyse d'un groupe d'images peut parfois être assez long et excéder une minute.

Implémentation

Comme dit précédemment l'avantage de l'API web est son faible besoin en ressources matérielles, il est donc possible de confier au robot l'exécution de ce module contrairement à la solution précédente qui nécessite de déporter les calculs sur un ordinateur externe. Cependant son instabilité en termes de temps de réponse, principalement due aux quotas imposés par la plateforme, rend cette solution peu viable à l'heure où est écrit ce rapport mais est susceptible de changer. Les avantages et inconvénients de ces deux méthodes sont résumées dans le tableau 5.1 ci-contre.

	Calculs	**Connectivité**	**Temps de traitement**
Implémentation locale	+	-	~5,8 sec
API Orange	-	+	~5 sec

Table 5.1 – Possibilités d'implémentation d'un algorithme de détection d'objets

A l'issue de cette étude j'ai finalement choisi de développer un module ROS pour chacune des deux solutions présentées afin de tirer profit de leurs avantages respectifs. Les deux modules héritent de la même classe mère et leurs contrats d'interface similaires les rendent interchangeables vis à vis de ROS.

5.2.4 Configuration du Dialogue

Le dialogue multimodal a été configuré pour les besoins du scénario, avec notamment le rajout d'une modalité d'entrée, la détection d'objet. Son principe est simple, dans un premier temps une image est prise à l'aide des caméras présentes sur le robot, puis les objets présents sont détectés à l'aide de Tensorflow. Dans un second temps, le module de modalité d'entrée envoie au moteur de dialogue la classe de l'objet ayant le score le plus

4. HTTP : HyperText Transfer Protocol

haut si il y a lieu.

Le dialogue a par la suite été configuré pour permettre une description orale de l'objet recherché par l'utilisateur. Ce cas apparaît pertinent car l'interlocuteur n'a pas systématiquement un objet équivalent sur lui, dans le cas où il cherche son téléphone par exemple il paraît raisonnable de supposer qu'il n'en a pas d'autre.

Conclusion

Mon stage s'est soldé par la mise en œuvre d'un dialogue multimodal avec un robot, objectif initial atteint grâce notamment à l'apport technique de mon tuteur, Jacques Chodorowski. Ce projet constitue une expérience gratifiante tant sur le plan personnel que professionnel. Techniquement il m'a permis d'acquérir une expérience considérable en Python et sur l'utilisation du middleware ROS. Des apports non techniques sont également à noter comme la méthodologie agile Kanban qui pourra être mise à profit dans mes travaux futurs.

Ce stage de fin d'études a finalement été pour moi une occasion exceptionnelle d'être plongé dans une équipe de recherche dans un secteur en pleine effervescence qui de plus est en parfaite adéquation avec mon cursus. Il m'a en outre permis de conforter mon projet professionnel : je souhaite en effet poursuivre ma carrière dans la recherche en robotique. Par ailleurs Orange m'a offert la possibilité de poursuivre les travaux effectués durant mon stage au cours d'une thèse, je suis particulièrement reconnaissant pour cela.

Ce sujet de thèse vise à combler les lacunes identifiées dans l'implémentation actuelle, d'une part le manque d'adaptabilité vis à vis de l'environnement du robot, ou vis à vis des utilisateurs. Il s'agit d'intégrer d'autre part les interactions multimodales complémentaires ignorées jusqu'ici. Ce projet ambitieux s'avère particulièrement intéressant et traite de problématiques très diverses.

Table des figures

Liste des Codes Sources

Liste des tableaux

Annexes

Annexe A : Configuration du Moteur de Fusion

Ce document détaille la structure JSON du fichier *input.json* visant à configurer le moteur de fusion du dialogue multimodal. Ce fichier décrit les différentes modalités d'entrée utilisées pour l'interaction ainsi que les actions pouvant être déclenchées.

Racine

Clé	Type	Requis	Défaut	Description
modalities	Modality[]	Oui	-	La liste des modalités d'entrée
triggers	Trigger[]	Oui	-	La liste des déclencheurs d'action
actions	Action[]	Oui	-	La liste des actions

Modality

Clé	Type	Requis	Défaut	Description
name	string	Oui	-	Le nom de la modalité, doit être unique
bargein	bool	Non	false	Si vrai, cette modalité peut interrompre le robot pendant son interaction, toutes les modalités de sortie sont ainsi stoppées

Trigger

Clé	Type	Requis	Défaut	Description
name	string	Oui	-	Le nom du déclencheur, doit être unique
source	string	Oui	-	Le nom de la modalité associée à ce déclencheur, doit correspondre exactement au champs « name » d'une modalité, le déclencheur sera sinon ignoré
value	string	Oui	-	La valeur conditionnant le déclenchement, sensible à la casse

Action

Clé	Type	Requis	Défaut	Description
name	string	Oui	-	Le nom de l'action, doit être unique
timeWindow	float	Oui	-	La fenêtre de temps en secondes pendant laquelle les déclencheurs sont considérés ensemble
triggers	string[]	Oui	-	La liste des combinaisons de déclencheurs de l'action, chaque élément de la liste correspond à une combinaison. Une combinaison peut contenir le nom d'un ou plusieurs déclencheurs séparés par le symbole && (les espaces sont ignorés)

Annexe B : Configuration du Moteur de Dialogue

Ce document détaille la structure JSON du fichier *machine.json* visant à configurer le moteur de dialogue du dialogue multimodal. Ce fichier vise à implémenter la logique de dialogue, ses états et ses transitions.

Racine

Clé	Type	Requis	Défaut	Description
states	State[]	Oui	-	La liste des états
transitions	Transition[]	Oui	-	La liste des transitions

State

Clé	Type	Requis	Défaut	Description
name	string	Oui	-	Le nom de l'état, doit être unique
on_enter	string	Non	-	Fonction Python appelée lorsque la machine à états transite vers cet état. A implémenter dans *DialogueEngine*, la fonction prend un paramètre *action*
on_exit	string	Non	-	Fonction Python appelée lorsque la machine à états transite depuis cet état vers un autre. A implémenter dans *DialogueEngine*, la fonction prend un paramètre *action*

Transition

Clé	Type	Requis	Défaut	Description
source	string	Oui	-	Le nom de l'état de départ
trigger	string	Oui	-	Le nom de l'action déclenchant la transition, il doit correspondre à une action valide du moteur de fusion. Peut prendre comme valeur « * », dans ce cas la transition sera prise si aucune action n'est identifiée lors d'une interaction utilisateur
dest	string	Oui	-	Le nom de l'état de destination
conditions	string	Oui	-	Fonction Python appelée avant d'effectuer la transition, cette dernière est effectuée seulement si la fonction renvoie *True*. A implémenter dans *DialogueEngine*, la fonction prend un paramètre *action*
before	string	Oui	-	Fonction Python appelée avant d'effectuer la transition. A implémenter dans *DialogueEngine*, la fonction prend un paramètre *action*
after	string	Oui	-	Fonction Python appelée après avoir effectué la transition. A implémenter dans *DialogueEngine*, la fonction prend un paramètre *action*

Annexe C : Configuration du Moteur de Fission

Ce document détaille la structure JSON du fichier *output.json* visant à configurer le moteur de fission du dialogue multimodal. Ce fichier décrit les modalités de sortie ainsi que les interactions du robot.

Racine

Clé	Type	Requis	Défaut	Description
modalitites	Modality[]	Oui	-	La liste des modalités de sortie
outputs	Output[]	Oui	-	La liste des interactions

Modality

Clé	Type	Requis	Défaut	Description
name	string	Oui	-	Le nom de la modalité, doit être unique
topic	string	Oui	-	Le nom du topic ROS associé à la modalité, ce topic doit correspondre à un *actionlib server* de type OutputAction

Output

Clé	Type	Requis	Défaut	Description
action	string	Oui	-	Le nom de l'action, doit être unique et doit correspondre à une action du moteur de fusion ou une valeur fixée par une routine du moteur de dialogue
[MODALITY]	string string[] ModOutput	Non	-	La clé doit correspondre à un ou plusieurs noms de modalité séparés par le symbole &&. Le contenu sera envoyé à(aux) interactions concernée(s). Si la valeur est un string, celui-ci sera renseigné dans le champs text de l'OutputGoal. Si la valeur est une liste de string, un élément sera choisi aléatoirement et renseigné dans le champs text de l'OutputGoal. Sinon, la valeur doit être un objet ModOutput valide

ModOutput

Clé	Type	Requis	Défaut	Description
text	string string[]	Oui	-	Si la valeur est un string, celui-ci sera renseigné dans le champs *text* de l'OutputGoal. Si la valeur est une liste de string, un élément sera choisi aléatoirement et renseigné dans le champs *text* de l'OutputGoal
data	string[]	Non	-	Liste de chaînes de caractères

Bibliographie

[1] N. Sebe A. Jaimes. Multimodal human-computer interaction : A survey. 2007.

[2] Richard A. Bolt. *"Put-that-there" : Voice and gesture at the graphics interface*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1980.

[3] Joëlle Coutaz. *Four easy pieces for assessing the usability of multimodal interaction : the CARE properties*. PhD thesis, Laboratoire de Génie Informatique, Grenoble, France, 1995.

[4] Bruno Dumas. *Frameworks, description languages and fusion engines for multimodal interactive systems*. PhD thesis, Université de Fribourg, Suisse, 2010.

[5] Bruno Dumas. Fusion in multimodal interactive systems : An hmm-based algorithm for user induced adaptation. 2012.

[6] Søren Tranberg Hansen. A multimodal robot game for seniors. 2017.

[7] Juliette Paoli. Spoon, le robot empathique et évolutif. *Solutions Numériques*, 2016.

[8] Jerry Carter Paolo Baggia, Daniel C. Burnett. Emma : Extensible multimodal annotation markup language. Technical report.

[9] Dennis Perzanowski. Building a multimodal human–robot interface. 2001.

[10] Ahmed Hussain Qureshi. Robot gains social intelligence through multimodal deep reinforcement learning. 2017.

[11] Vivek Rathod. *Tensorflow detection model zoo*.

[12] Shankar T. Shivappa. Audiovisual information fusion in human-computer interfaces and intelligent environments : A survey. 2010.