UNIVERSITÉ DU QUÉBEC À MONTRÉAL

RECONNAISSANCE D'ACTIVITÉS ET RECONNAISSANCE D'OBJETS PAR AFFORDANCE POUR LA RECONNAISSANCE DE PLAN D'UN ROBOT ASSISTANT COGNITIF

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INFORMATIQUE

PAR

MATHIEU GRAVEL

JUIN 2019

REMERCIEMENTS

Dans un premier temps, je tiens à remercier mon directeur de recherche, le professeur Éric Beaudry, qui m'a offert de l'aide et des réponses à mes questions tout au long de la maîtrise, ainsi que des possibilités uniques dans le cadre de la recherche.

Je souhaite ensuite remercier Jean Massardi, étudiant au doctorat, avec qui j'ai collaboré tout au long de maîtrise. En effet, le module de reconnaissance d'objets et d'actions que j'ai développé produit en sortie l'entrée du module de reconnaissance de plans qui développe. Ses travaux m'ont aidé à mieux orienter les miens et je lui en suis reconnaissant.

J'aimerais aussi remercier Jaël Champagne Gareau, Éric Lavallée, Philippe Billard, Guillaume Gosset et Gabrielle Hunter, pour leurs aides diverses allant de l'annotation d'images à l'aide pour porter le code Windows sur Linux.

Je remercie le Fonds de recherche du Québec — Nature et technologies (FRQNT) et le Conseil de recherches en sciences naturelles et génie (CRSNG) pour leur soutien financier. Je remercie également le réseau d'excellence pancanadien AGE-WELL pour l'opportunité unique d'avoir pu travailler sur ce projet, pour l'expérience enrichissante acquise lors de la conférence AGE-WELL 2018 à Vancouver et pour son soutien financier. Je souhaite aussi remercier la Faculté des sciences de l'UQAM pour son soutien académique.

Finalement, je remercie ma famille et mes proches, qui m'ont offert un support moral tout au long de mes études.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

RÉSUMÉ

Le vieillissement de la population et le besoin de soutien grandissant des personnes âgées dans leur vie quotidienne sont un enjeu sociétal actuel au Canada. C'est pour répondre à cette problématique que le réseau AGE-WELL finance un projet de conception d'un robot assistant cognitif pour aider une personne âgée à la réalisation de ses activités quotidiennes. Or, ce type de système doit comporter un module de vision et de reconnaissance d'activités afin de comprendre les actions qu'une personne est en train d'exécuter. Cette information est d'une importance capitale pour comprendre ses plans et ses intentions, afin de l'aider à les réaliser. Nous proposons dans ce mémoire une approche de reconnaissance d'activités par la reconnaissance d'interactions humains-objets, grâce à une représentation d'activités en tant qu'affordances. Pour ce faire, nous avons développé le module PARC, Plan and Activity Recognition Component. PARC est conçu pour reconnaitre les interactions d'une personne avec un objet, afin de modéliser ses activités sous ressources limitées et de les transmettre à un système de reconnaissance de plan, le tout dans le but d'être intégré à une plateforme robotique. Ce mémoire se concentre sur la partie de reconnaissance d'activités. Celle-ci fonctionne par la détection d'affordances entre un humain et un objet, qui sont eux-mêmes détectées par la classification d'objets d'un réseau de neurones. Afin d'assurer que PARC puisse améliorer son ensemble de connaissances spécifiques à l'utilisateur et aux objets de son environnement, il est aussi doté d'un système d'auto-apprentissage lui permettant d'ajouter à son système de détection d'objets la capacité de détecter de nouvelles classes et de nouvelles représentations de classes d'objets existants. L'approche proposée a été évaluée à l'aide d'un corpus spécifique créé pour l'occasion, ainsi que sa librairie de plans proposés. L'évaluation du module démontre son efficacité à déceler les objets de son environnement ainsi que les activités effectuées avec cesdits objets, pour les transmettre à un système de reconnaissance de plan de haut-niveau. Le système démontre aussi une amélioration de son efficience à long terme, grâce à un processus d'auto-apprentissage sur les données cachées de son environnement, qui se déroule parallèlement à la tâche de reconnaissance d'activités.

MOTS-CLÉS : vision par ordinateur, vision artificielle, reconnaissance d'objet et d'activité, apprentissage machine, intelligence artificielle, robotique mobile.

INTRODUCTION

Le Canada, tout comme d'autres pays développés, fait face au vieillissement de sa population. L'augmentation relative de population âgée par rapport à la population active (en âge de travailler) (Canada, 2012) ramène au premier plan plusieurs enjeux sociétaux, comme l'accès aux services de santé et sociaux et à leur financement (Paez *et al.*, 2010). Cela a aussi remis en question l'âge de la retraite (Fougère *et al.*, 2009).

Pour maintenir la qualité de vie des personnes âgées et limiter la pression des coûts du système public de santé, une approche qui fait consensus est de favoriser l'autonomie des personnes âgées afin qu'elles puissent rester le plus longtemps possible à domicile. C'est dans cette perspective que le réseau d'excellence pancanadien AGE-WELL a été créé. AGE-WELL a pour but de soutenir financièrement des projets de recherche appliquée dédiés au développement de technologies et de services destinés aux personnes âgées.

L'un des projets financés par AGE-WELL est le projet *Vigil : Mobile Robots for Telepresence and ADL (Activities for Daily Living) Assistant.* Ce projet, mettant en collaboration des équipes de l'Université de Sherbrooke, de l'Université de Toronto, de l'Université de Western Ontario et de l'Université du Québec à Montréal (UQAM), a pour objectif de concevoir un robot assistant cognitif capable d'accompagner des personnes âgées dans leurs activités quotidiennes à domicile.

Les robots-assistants peuvent être catégorisés selon le type d'aide qu'ils offrent, tels que les robots compagnons qui offrent un support psychologique à la personne ou les robots de type service, qui offrent un support physique afin de garder leur

autonomie (Broekens *et al.*, 2009). Une des sous-catégories de ce type d'agent robotique est celle des robots assistants cognitifs. Avec ce type de robot assistant, on veut résoudre la problématique de reconnaitre les habitudes et activités quotidiennes d'une personne et ce, pour limiter les oublis possibles lors de leur traitement. Certaines recherches des dernières années dans ce domaine montrent que ce type de robots offre une aide similaire à celle des programmes d'entrainement cognitif et permet la stimulation du cerveau en vue de prévenir ou d'atténuer le déclin cognitif (Robinson *et al.*, 2014).

Dans le cadre du projet *Vigil*, l'équipe de l'UQAM a pour mandat de réaliser un module de reconnaissance d'activités et de plan. Cette détection se fait à partir d'images captées par une caméra installée sur le robot assistant.

Afin de concevoir un tel robot pouvant offrir une aide cognitive utile, diverses capacités d'intelligence artificielle doivent être implémentées et intégrées. Ce mémoire porte sur la mise en œuvre de capacités de reconnaissance d'objets dans son environnement, la considération du contexte, et la reconnaissance d'actions faites de la personne assistée qui sont liées à des objets déterminés. Ces capacités de reconnaissance sont essentielles au bon fonctionnement du robot assistant, car d'autres modules, comme la reconnaissance de plans, nécessitent la compréhension de l'environnement et des actions effectuées par la personne assistée. Il existe donc un besoin critique d'une approche pour reconnaître les actions qu'une personne peut entreprendre lors de l'exécution de ses activités quotidiennes ainsi que des moyens de détecter les objets qu'elle utilise pour les réaliser.

On peut trouver dans l'état de l'art plusieurs avancées pertinentes pour la reconnaissance d'activités d'une personne, tel que déceler ses mouvements possibles futurs afin de mieux la suivre (Dutta et Zielinska, 2017), améliorer les données traitables par l'ajout de divers capteurs infrarouges ou RFID dans l'environnement

(Arcelus *et al.*, 2007; Gross *et al.*, 2011) ou par l'utilisation de connaissances à priori, afin de générer les plans et les connaissances de l'environnement (Schroeter *et al.*, 2013). Pour ce qui est du domaine de la vision par ordinateur, la plupart des approches modernes utilisent des techniques d'extraction de caractéristiques et d'apprentissage profond (Zheng *et al.*, 2017) afin de déterminer l'emplacement d'un objet dans une image ainsi que sa classification (Ren *et al.*, 2015; Gidaris et Komodakis, 2015; He *et al.*, 2017).

Ces approches, quoiqu'efficaces dans leur contexte particulier, présentent toutefois certaines lacunes lorsqu'elles sont appliquées en robotique, comme dans le cas du robot assistant visé par notre projet. Elles présentent de plus une difficulté à équilibrer la qualité, l'amélioration des résultats et le temps d'entraînement et de calcul (Ren *et al.*, 2015), les méthodes modernes se basent souvent sur une base de connaissances pré-entrainée et prennent peu souvent en compte l'apparition de nouveaux objets dans l'environnement comme valeurs d'amélioration de leur système de connaissance. Ces plateformes d'assistants robotiques peuvent aussi présenter plusieurs limitations physiques à prendre en compte lorsque la plateforme robotique ne détient aucun capteur de données de l'environnement autres que ceux placés sur le robot lui-même.

L'objectif principal de ce mémoire est de présenter un module de traitement de vision par ordinateur et de reconnaissance d'activités intégrable à un robot assistant. Plus particulièrement, on veut démontrer grâce au système proposé qu'il est possible de bâtir un système de vision pour la reconnaissance d'activités qui intègre les forces des approches modernes en apprentissage machine, et cela, avec la possibilité de s'auto-améliorer et de détecter de nouveaux objets hors de sa base de connaissances. Ce système doit aussi fonctionner avec des ressources limitées, afin de faciliter son intégration dans un robot assistant qui doit aussi faire fonctionner plusieurs autres modules critiques en parallèle.

Le module/système de reconnaissance d'objets et d'actions présenté s'intègre à d'autres modules du robot, dont un module de reconnaissance de plan et de prédiction d'actions programmé par un autre étudiant sur le projet. La séquence d'activités reconnus est envoyée à cet autre module afin d'estimer le plan actuel de la personne assistée et d'anticiper ses actions futures, et ce, dans le but de lui fournir une aide utile.

Le module de reconnaissance d'objets utilise un système de vision par réseaux de neurones à convolution pour détecter et classifier les objets connus captés par une caméra. Afin de détecter de nouvelles formes d'objets, nous avons conceptualisé l'algorithme *Selective Search Depth* afin de détecter de nouvelles classes d'objets précédemment inconnu par le système afin de pouvoir les étiqueter. Ces données sont ensuite normalisées selon le concept d'affordance, afin de détecter une action en cours. L'approche a été expérimentée avec un corpus de plans et d'activités. Les résultats obtenus démontrent la pertinence de l'approche proposée. En effet, le système a démontré une forte capacité à s'adapter aux différents plans et une robustesse à la présence de bruit dans l'environnement.

Le présent mémoire est divisé en cinq chapitres. Le premier chapitre présente les approches traditionnelles et actuelles de vision par ordinateur et les différentes problématiques associées à la recherche d'un objet dans une image. Le deuxième chapitre continue cette problématique et met l'accent sur les approches basées sur l'apprentissage profond qui ont gagné en popularité ces dernières années. Le troisième chapitre examine la reconnaissance d'activités et de plan, introduit le concept d'affordance et ses utilisations pour déterminer les liens objet-action. Le quatrième chapitre présente le module de reconnaissance d'activité par reconnaissance d'affordance, sa méthodologie, son intégration et son fonctionnement avec un système de reconnaissance de plan. Enfin, le cinquième chapitre reporte et analyse les résultats des expérimentations effectuées. Une conclusion clôt le mémoire.

CHAPITRE I

VISION PAR ORDINATEUR

La vision par ordinateur est un domaine de recherche qui vise à concevoir des systèmes capables de traiter des images ou des séquences d'images (vidéos) afin d'en extraire des informations pertinentes. Ce domaine, aussi connu sous le nom de vision artificielle, est généralement considéré comme un sous-domaine de l'intelligence artificielle (IA). Les systèmes de vision artificielle mettent en œuvre diverses fonctions et méthodes, comme des algorithmes de prétraitement, de classification, de segmentation, d'apprentissage machine, etc., afin d'extraire des informations des zones d'intérêts dans l'image. Puisque la classification s'effectue sur ces zones d'intérêts, il est essentiel d'extraire des espaces riches en informations.

Par exemple, un être humain peut aisément déterminer que les trois images de la figure 1.1 représentent le même objet, soit une tasse. Même si la seconde image de la tasse détient des données visuelles superflues (l'ajout d'un X) et que la troisième image a subi une rotation de 180 degrés et est devenue plus transparente, notre cerveau est toujours capable de les reconnaître comme une copie de l'objet d'origine. Le cerveau humain y arrive grâce à l'utilisation d'entités visuelles par notre système neurocognitif ; des représentations abstraites de surfaces et d'objets qui permettent de les identifier même si la forme analysée présente des différences comparées à la base des connaissances visuelles de la personne (Cavanagh, 2011).

Figure 1.1: Représentation d'images avec occlusions.

Pour un être humain, une entité visuelle est bâtie durant l'entièreté de sa vie selon un grand nombre de critères, tels que l'environnement qu'il perçoit, les scènes ou objets perçus, les différentes méthodes d'interaction entre la personne et l'objet, etc. Cette complexité rend le système de vision humaine actuellement impossible à implémenter dans son intégralité. Il est donc nécessaire de déterminer quelles sont les informations les plus riches en données pertinentes.

Dans le reste de ce chapitre, nous définissons ce que sont plus exactement les zones d'intérêts à analyser dans une image et les méthodes courantes pour les détecter. Nous présentons ensuite comment ces zones d'intérêts peuvent servir à reconnaitre de manière fine des formes dans une image. Enfin, nous présentons les méthodes de vision par ordinateur et d'intelligence artificielle utilisées pour effectuer ces détections d'objets.

1.1 L'analyse d'une image

Comme son nom l'indique, une image numérique est une représentation numérique, qui prend souvent la forme de matrices à dimensions $M \times N$, où M et N sont la hauteur et largeur d'une image. La matrice d'une image contient, pour chaque emplacement interne, une structure permettant de représenter l'intensité moyenne de la lumière à cet endroit que l'on appelle un **pixel**. Un pixel n'est toutefois pas une représentation parfaite de l'information visuelle de ce qu'elle représente, puisque la limitation des capteurs cause une perte d'information.

En effet, différents facteurs peuvent influencer la perception de la couleur et de la forme d'un objet, tels que la quantité de lumière atteignant l'objet, la réflexion à sa surface ou même la qualité du système de capture d'image pour l'acquisition des données.

Cette perte possible d'information est exacerbée lors de l'utilisation de capteurs numériques. Le système de vision du cerveau humain s'appuie sur la moitié du cortex du cerveau (Wördenweber *et al.*, 2007), soit environ 30 milliards de neurones et plusieurs trilliards de synapses, et utilise différents systèmes encore peu connus par les chercheurs, définis par divers modèles opposés (Gibson, 1966; Gregory, 1970) afin d'enrichir leurs données par les informations déjà connues, telles que sa représentation 3D, ses textures, ses motifs et bien plus encore.

Une caméra ne peut offrir seule ce processus et transmet donc une perception subjective incomplète de l'environnement perçu. Chaque pixel de l'image peut alors seulement représenter l'intensité moyenne de luminance pour son emplacement, nommé $I_{x,y}$, et toute information supplémentaire nécessite l'utilisation de plusieurs systèmes additionnels pour les enrichir.

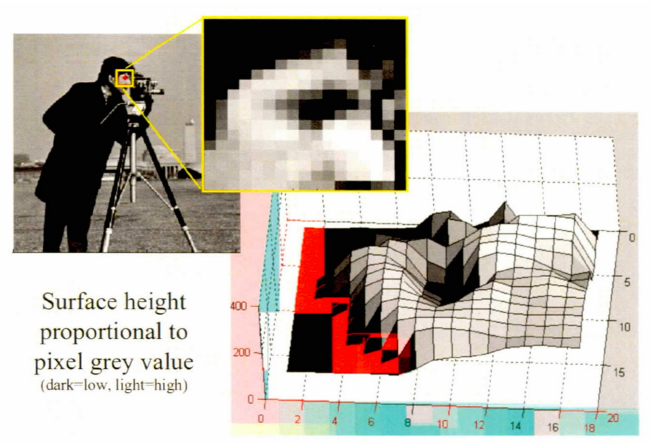


Figure 1.2: Représentation des intensités de couleurs pour une image. Source : Robert Collins, Université de Penn State [1]

.

La représentation interne des images en matrices 2D cause une autre limitation, c'est-à-dire que les pixels contenus représentent seulement une intensité de luminance, généralement sur une échelle entre 0 et 255 (8 bits). Puisque cette structure limiterait la quantité de couleurs entreposable, il est nécessaire de stocker les différentes intensités en plusieurs **canaux**. Un canal, dans le contexte de l'imagerie numérique, est un regroupement distinct d'un ensemble de valeurs associées à une image. Par sa nature, un canal peut emmagasiner n'importe quelle information visuelle distincte d'une image, que ce soit son taux d'intensité d'une couleur, son niveau de transparence, et bien plus.

En général, les canaux d'une image sont utilisés pour entreposer les **espaces de couleur**, c'est-à-dire les canaux de luminance basés sur différentes représentations de couleurs primaires. L'utilisation de multiples canaux pour un pixel permet ainsi de former la couleur spécifique du pixel, tel que les canaux **RGB**.

1. `http://www.cse.psu.edu/~rtc12/CSE486/`

1.2 Le traitement d'image

1.2.1 L'analyse numérique

Une des méthodes les plus couramment utilisées pour la détection d'objets est *la comparaison avec des motifs standards*, c'est-à-dire le calcul de la distance de diverses zones de l'image avec un modèle de l'objet qui est recherché. Puisqu'un objet peut être détecté sous différentes formes, tailles et orientations dans l'image analysée, une classification naïve devrait considérer toutes les sous-matrices possibles de la matrice d'origine, ainsi que toutes les transformations géométriques possibles (rotation, homothétie, changement de contraste, etc.).

Malheureusement, la quantité de calculs nécessaire pour cette approche naïve augmente exponentiellement en fonction de la taille de l'image. Prenons pour exemple une image de 10 par 10 pixels. Une analyse complexe portée sur cette minuscule image ainsi que l'analyse de toutes les sous-matrices adjacentes possibles de pixels de taille,

$$1^2 + 2^2 + \cdots + n^2 = \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} = \mathbf{O}(n^2) \tag{1.1}$$

, et leurs transformations par rotation entre 0 et 360 degrés, nécessiterait une classification sur $\frac{10\times11\times21}{6} \times 360 = 138\,600$ sous-matrices, afin de reconnaître toutes les variantes possibles de l'objet. Cela demanderait une trop grande capacité de calcul. Il est donc nécessaire d'utiliser des techniques d'extraction d'informations de l'image plus substantielles, afin de limiter la portée de la classification et la perte d'information analysable, afin de générer ce qu'on appelle des **régions d'intérêts**.

Dans la littérature, les régions d'intérêts sont utiles pour focaliser les algorithmes d'extraction de **caractéristiques** (*features* en anglais). On utilise ce terme en imagerie numérique pour décrire l'ensemble des données qui permettent de décrire des informations pertinentes sur les différents objets qui peuvent être identifiés sur une

image. Une caractéristique peut être de différentes tailles, formes, et peut être soit uniue, soit partagés entre différents objets. Il existe pour un objet autant de caractéristiques que d'observations théoriques possibles sur ces détails. Par exemple, une image composé de 4 carrés de couleurs différents seulement détiendrait des caractéristiques de couleur **RVB** sur les pixels, une caractéristique de dimension des carrés, une caractéristique de contours décelables par l'intersection des quatre coins et bien plus encore.

Un des facteurs importants des caractéristiques est ce qu'on appelle leur **taux de variance (ou d'invariance)** vis-à-vis des facteurs de changements des données visuelles, tels que les transformations géométriques ou les changements de luminosité. Par taux de variance, on décrit les facteurs qui peuvent changer l'apparence d'un environnement temporairement sans toutefois changer son contenu.

Lors de l'extraction des caractéristiques, les algorithmes de traitement d'images de bas niveau vont travailler selon les matrices sous-jacentes des pixels. Certains facteurs lors de l'acquisition de l'image peuvent changer la valeur des matrices sous-jacentes de manière assez significative de telle sorte qu'elles ne ressemblent plus à leurs formes d'origine. Ces limitations en traitement d'image poussent ainsi l'utilisation de caractéristiques que l'on nomme haut-niveau, c'est-à-dire une représentation de caractéristiques mathématiques invisibles à l'œil nu, qui sont invariantes à ces facteurs d'incertitudes. Des exemples de l'effet de différents facteurs de variances appliqués à la même image d'origine sont montrés aux tableaux 1.1 et 1.2. Ces tableaux montrent une image ainsi qu'une représentation matricielle de ses pixels de taille 5×5, toujours extraits de la même position dans l'image d'origine.

Plusieurs approches de génération de caractéristiques invariantes à ces transformations ont été proposées, telles que la détection de lignes, d'ellipses, etc. Plusieurs

ont proposé des approches haut-niveau afin de générer des zones d'intérêts invariantes à plusieurs caractéristiques, qui vont être décrites plus en détail à la section 1.3. Ces approches, ainsi que leur utilité pour les différentes facettes de classification en imagerie numérique, sont présentées à la prochaine section.

Tableau 1.1: Représentation de différents facteurs de variance partie 1

Facteur de variance	Exemple	Exemple de matrice
Image de base		[80 73 57] [79 72 56] [79 72 56] [82 70 58] [82 70 58] [81 73 60] [80 72 59] [78 70 57] [82 69 60] [83 70 61] [82 74 61] [80 72 61] [78 70 59] [83 70 61] [83 70 62] [77 73 62] [79 75 64] [81 77 68] [74 68 68] [74 69 66] [80 73 63] [83 76 68] [84 77 69] [76 67 68] [77 69 67]
Luminance		[125 110 89] [113 98 77] [116 101 80] [124 109 88] [129 114 93] [125 110 89] [117 102 81] [122 107 86] [125 110 89] [133 118 97] [122 107 86] [121 106 85] [131 116 95] [138 123 102] [146 131 110] [145 130 109] [146 131 110] [146 131 110] [140 124 108] [140 124 108] [135 120 99] [138 123 102] [140 125 104] [131 115 99] [132 116 100]
Angle		[97 89 68] [99 90 75] [101 91 81] [95 93 81] [91 89 77] [98 90 69] [101 92 77] [102 92 82] [93 91 78] [91 89 76] [99 91 70] [101 92 77] [103 93 83] [90 88 75] [91 89 76] [93 93 69] [92 91 71] [89 91 70] [95 87 84] [94 85 86] [92 91 70] [92 91 71] [90 91 73] [101 93 90] [97 88 89]

Tableau 1.2: Représentation de différents facteurs de variance partie 2

Facteur de Variance	Exemple	Exemple de Matrice
Changement de taille		[72 53 38] [72 53 38] [73 54 39] [69 51 31] [69 51 31] [72 53 38] [73 54 39] [73 54 39] [69 51 31] [69 51 31] [73 54 39] [73 54 39] [74 55 40] [69 51 31] [69 51 31] [70 54 38] [71 55 39] [71 55 39] [71 56 35] [71 56 35] [70 54 38] [71 55 39] [71 55 39] [71 56 35] [71 56 35]
Changement de contraste		[1 1 1] [1 1 1] [1 1 1] [1 1 1] [1 1 1] [1 1 1] [1 1 1] [1 1 1] [1 1 1] [1 1 1] [1 1 1] [1 1 1] [1 1 1] [1 1 1] [1 1 1] [0 3 0] [0 1 0] [1 0 0] [1 1 1] [1 1 1] [0 2 0] [0 0 0] [4 0 1] [1 1 1] [1 1 1]
Occlusions		[94 85 80] [94 85 80] [94 85 80] [93 84 79] [95 86 81] [95 86 79] [97 88 81] [98 89 82] [90 81 74] [93 84 77] [95 86 77] [98 89 80] [100 91 82] [88 79 70] [91 82 73] [89 80 71] [87 79 68] [86 78 65] [82 76 54] [85 79 57] [89 80 71] [87 79 68] [85 77 64] [83 77 55] [85 79 57]

Un autre problème courant pour la création de zones d'intérêt est le traitement des caractéristiques de couleur. Les espaces de couleur habituels, tels que *Rouge-Vert-Bleu* ou *YUV*, découpent les couleurs primaires en trois canaux séparés. Par défaut, les structures ne peuvent accorder aisément d'informations aux mélanges des canaux et nécessitent des transformations séparées, telles que l'utilisation d'histogrammes de distribution, afin de détecter et de transformer les zones de pics d'intensité (Van de Weijer *et al.*, 2006; Sural *et al.*, 2002).

1.3 Classification numérique

1.3.1 Extraction de caractéristiques

On a fait remarquer à la section précédente le besoin d'utiliser des techniques avancées afin de pouvoir analyser les zones d'intérêt d'une image couleur. Il est important d'extraire les zones riches en informations afin de déceler les caractéristiques uniques aux différentes formes reconnaissables d'une image. On peut alors utiliser ces caractéristiques extraites comme points clés de représentation des divers objets afin de pouvoir les détecter et les classifier plus aisément. En fait, ils sont la base sur laquelle reposent les trois catégories d'algorithmes de classification en imagerie numérique, c'est-à-dire la **reconnaissance de formes**, la capacité de détecter une zone prometteuse et riche en information ; la **segmentation de formes**, la capacité de segmenter cette zone selon ses différentes formes ; et la **classification de formes**, la capacité de classifier par techniques d'apprentissage machine les caractéristiques de la zone selon différentes catégories.

On souhaite donc l'utilisation des techniques rapides et efficientes, afin de déceler le plus de caractéristiques riches en informations utilisables pour ces différentes catégories.

La première étape d'extraction de caractéristiques consiste à déterminer quelles informations d'une image sont pertinente et quelles sont celles qui peuvent être perçues comme du **bruit**. On considère le bruit comme étant les caractéristiques de variation de luminance ou de couleur d'une image introduites par erreur par les capteurs. Ces valeurs ne sont pas représentatives de la source d'origine et peuvent causer des cas d'erreurs. La source d'un bruit peut venir de différentes choses, telles que la qualité du capteur choisi pour numériser l'image, les algorithmes de compression utilisés, etc.

La figure 1.3 présente l'effet d'addition de bruit sous format *poivre et sel* sur une

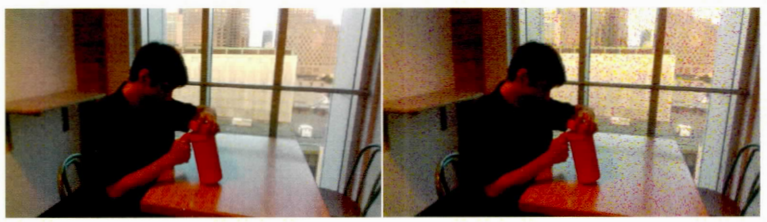


Figure 1.3: Exemple de bruit dans une image.

image. En plus de causer une perte d'information, le bruit détient un facteur de variation élevé comparé au reste de pixels de l'image et peuvent avoir un effet nocif sur la détection. Il est donc nécessaire d'effectuer un processus de nettoyage sur les données de l'image avant leur utilisation dans des algorithmes de détection et de classification des formes. Pour ce faire, on peut effectuer un lissage des détails de l'image afin de garder seulement les informations essentielles.

Une autre méthode possible pour la prise en compte du bruit est l'utilisation de techniques d'extraction de caractéristiques avec différents taux d'invariance au bruit. Toutefois, il n'existe aucune technique résistante aux bruits en tout temps. Le bruit d'une image peut être tellement significatif qu'il y entraîne une perte totale des données. Si on prend un cas extrême, il est impossible de lisser le bruit d'une image où l'intégralité des pixels a été convertie à la valeur $(0, 0, 0)$.

Une des approches de base afin de filtrer les détails erronés est l'utilisation d'un **filtre à convolution**, aussi appelé **masque**, **matrice de masquage** ou **matrice noyau**. En mathématique, la convolution est l'utilisation de deux fonctions, par exemple $f(x)$ et $g(x)$, afin d'en créer une troisième, $f*g(x) = \int f(y)g(x-y)dy$, qui représente l'effet de la première fonction sur la seconde. En imagerie numérique, la convolution nécessite une double intégrale afin de traiter les deux dimensions $hauteur \times largeur$ de l'image et la formule doit être convertie sous forme discrète. Ceci nous donne ce que l'on appelle un **masque de convolution spatiale**, qui

peut finalement être inversé afin d'être appliqué sur des images en entrée. Ceci nous donne la formule $F(x.y) = \sum_i \sum_j f(x+i.y+j)h(i.j)$. qui peut être appliquée à la matrice de pixels d'une image par multiplication de la matrice du noyau avec chaque sous-matrice possible de même taille dans l'image.

Le résultat d'un filtrage par la matrice noyau d'une image permet la diminution de la quantité de bruit. mais peut causer une perte de détails de petit niveau. Cette image est plus floue que celle d'origine. mais est plus résistante à l'extraction de faux positifs dus au bruit lors de la détection et de la classification des formes. Les filtres à convolution peuvent varier en taille et en valeurs. dépendamment de la portée du projet et de la qualité des images à traiter. Elles sont aussi utilisées dans d'autres algorithmes du domaine. tels que la classification par réseau neuronal convolutifs. qui est discuté au chapitre 2.

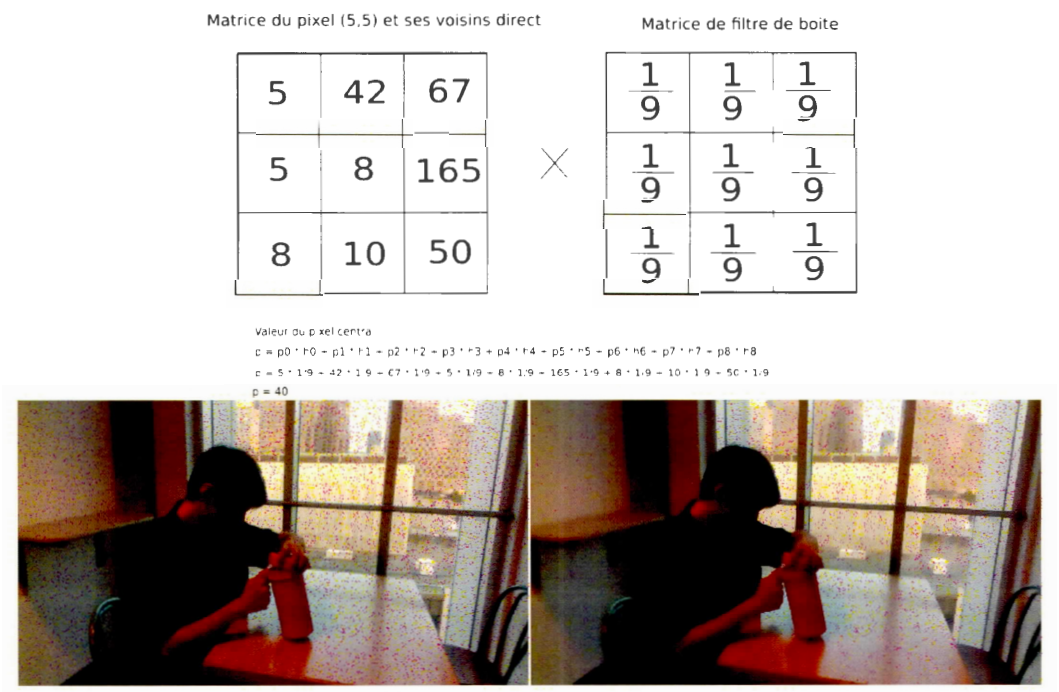


Figure 1.4: Exemple d'application d'une multiplication d'un pixel d'image par un noyau de filtrage et résultat final du filtrage sur la figure 1.3.

D'autres approches plus complexes existent pour le nettoyage du bruit dans des images. (Portilla *et al.*, 2003) propose de découper l'image en représentation pyramidale. Il s'agit de copier répétitivement l'image d'origine et de grossir linéairement les copies, pour ensuite calculer les coefficients de chaque région et leurs équivalents dans les images d'échelle proche afin de maximiser le nettoyage du bruit tout en limitant la perte d'informations. Une autre approche est celle de (Zhang *et al.*, 2017), qui propose d'utiliser un réseau de neurones profonds pour apprendre au système à discriminer entre les données pertinentes et celles causant du bruit. Ces approches offrent de meilleurs résultats que ceux de l'application de filtres de convolution simple, mais peuvent être excessives pour la portée du projet.

La prochaine étape après la phase de filtrage est l'extraction des caractéristiques. Comme décrit dans la section 1.2.1, on cherche à extraire les caractéristiques ayant le plus grand taux d'informations discriminantes invariantes. Certaines caractéristiques peuvent être aussi simples que l'extraction des contours et des coins d'une image par l'utilisation de **gradients**, des dérivés partiels appliqués sur les sous-matrices de l'image afin de déterminer l'orientation des pixels. Cette approche est utilisée par plusieurs algorithmes de base en imagerie tels que (Harris et Stephens, 1988). D'autres sont de plus haut niveau, abstraits et nécessitent des calculs plus complexes, comme les caractéristiques SIFT (Lowe, 2004) qui utilisent des différences gaussiennes entre l'image et ses versions floutées afin de déterminer les points d'intérêts invariants au changement d'échelle. Il existe plusieurs autres approches d'extractions de caractéristiques similaires, telles que SURF (Bay *et al.*, 2008) ou ORB (Rublee *et al.*, 2011) pour l'extraction de caractéristiques avec invariance aux changements de taille, d'angle de détection et de luminosité.

L'utilisation de ce type de caractéristiques, c'est-à-dire basées sur des concepts globaux en imagerie et non sur l'ensemble des données à analyser, peut donner de

bons résultats, mais ils ont une lacune majeure. Les caractéristiques extraites se basent sur des formules mathématiques abstraites à problématiser et ne prennent donc pas en compte plusieurs niveaux de détails spécifiques à l'ensemble d'images en entrée, tels que les relations spatiales entre diverses caractéristiques ou leurs liens hiérarchiques. Des données sont alors perdues lors de l'étape de la classification. Un exemple simple, mais substantiel de détails perdus est les caractéristiques de couleurs complexes, puisque ces algorithmes opèrent leurs méthodes d'extraction sur des canaux individuels. Une autre lacune importante est que ce type de système est agnostique au contexte, c'est-à-dire qu'il ne prend pas en compte le positionnement des objets à détecter dans l'image analysée. Ceci peut amener pour conséquence à la détection de faux positifs, puisque les caractéristiques extraites sont globales à l'image et n'ont pas de moyen de démarquer différentes régions d'objets, tel que représenté dans la figure 1.5.

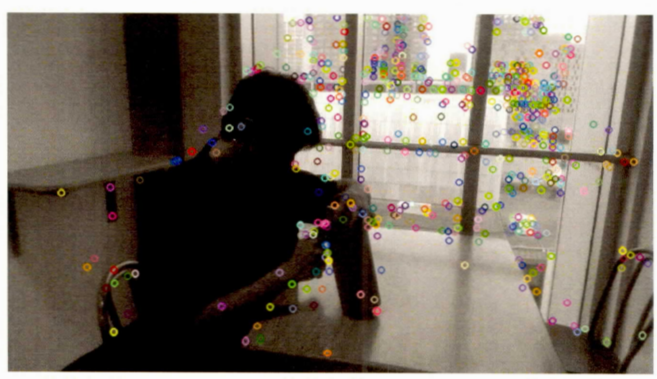


Figure 1.5: Caractéristiques de l'algorithme SIFT. L'algorithme détecte plusieurs caractéristiques sous format de points d'intérêts pour les formes de la personne et de la fenêtre, mais n'a aucun moyen de les démarquer en deux zones distinctes.

Pour régler cette dernière problématique, on utilise des méthodes de **recherche de régions**. Ces algorithmes permettent de découper une image en diverses zones distinctes, afin de permettre de limiter l'espace d'état classifiable en diverses sous-zones de l'image avec un fort potentiel de contenance des objets.

Une des approches les plus connues afin de détecter des régions est l'approche dite de la **fenêtre mobile**. Elle est un algorithme de recherche utilisé dans plusieurs secteurs de l'informatique, tels que la réseautique avec le *Sliding Window Protocol*, l'algorithmie et l'intelligence artificielle. La fenêtre mobile est utilisée souvent sous différentes formes dans le domaine du traitement de l'image, comme pour le traitement de filtres convolutifs ou l'extraction de régions d'intérêts (ROI)(F.Felzenszwalb, 2010).

Comme montré dans la figure 1.6, la technique effectue un découpage de toutes les régions d'intérêts de taille $n \times n$ possible dans une image. Pour chaque région, la fenêtre mobile extrait ses caractéristiques, possiblement calculées à l'avance afin d'éviter une redondance des calculs et le transmet à un classificateur. Après avoir parcouru toutes les sous-matrices de taille $n \times n$ possibles de l'image, on calcule ensuite une pyramide d'images (Adelson *et al.*, 1984) qui est retransmise à la fonction. Cette action nous permet de classifier les entrées selon différentes tailles afin d'assurer de ne pas manquer des objets qui sont plus grands ou plus petits que la fenêtre mobile d'origine.

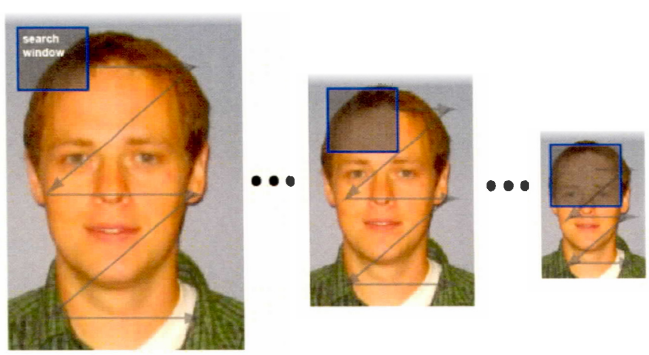


Figure 1.6: Exemple de fenêtre mobile.[2]

2. https://www.mathworks.com/help/vision/ref/vision.cascadeobjectdetector-system-object.html

C'est une approche classique dans le domaine, mais elle détient toutefois une déficience majeure au niveau du temps d'exécution, dû à la redondance des données dans les découpes. Certains chercheurs ont proposé des améliorations de l'algorithme, comme (Lampert *et al.*, 2009), qui propose d'utiliser une fonction de séparation et d'évaluation couplée avec une fonction d'évaluation générique, pouvant être implémentée par différentes métriques, telles que la fonction de décision d'un SVM ou la distance khi carré normalisé entre les histogrammes des sous-régions. Cette approche permet de limiter le nombre de sous-régions à évaluer, puisque l'algorithme préconise seulement la recherche dans les sous-ensembles avec un score égal ou supérieur à sa région majorante.

1.3.2 Génération de régions

Une autre approche existante pour le découpage d'une image en régions classifiables est la famille d'algorithmes de **génération de régions**. Ces algorithmes, similaires au fonctionnement des méthodes d'apprentissage non supervisé, inversent le raisonnement de l'approche de la fenêtre mobile. Plutôt que de découper des régions afin d'extraire des caractéristiques, on décèle des caractéristiques et des segments uniques dans l'image selon différents critères, tels que la similarité de couleur, de luminance, de profondeur, etc. et on utilise leur taux de rapprochements pour former des clusters. Cette approche se découpe principalement en deux catégories (Hosang *et al.*, 2016).

Algorithmes de regroupement

Les algorithmes de regroupement se basent sur le concept de génération de sous-régions. Pour ce faire, les algorithmes segmentent l'image par différents algorithmes de détection de contours, tels que le détecteur de Canny (Canny, 1987). Ces segments sont ensuite regroupés en régions probables selon leurs caractéris-

tiques uniques : les maximums de similarité de classe, la similarité de textures, etc. Ces approches ont un haut taux de précision, mais détiennent certains goulots d'étranglement de vitesse pendant la phase d'extraction de segments, dû à leur complexité élevée (Katiyar et Arun, 2014).

Image *Scoring*

Similaire à la technique de la fenêtre mobile, les algorithmes d'image scoring utilisent le principe de génération de régions sans connaissance à préavis dans l'image pour ensuite élaguer les candidats de réponses selon la qualité de leurs caractéristiques pour la classification. Une approche plus rapide que les algorithmes de regroupement, il détient néanmoins un taux de précision moins significatif. Ces algorithmes ont toutefois connu une certaine réémergence dans l'état de l'art des dernières années pour les classificateurs par apprentissage profondes.

Certains algorithmes dans ce groupe utilisent le principe de saillance, c'est-à-dire la mesure innée qu'aurait une chose pour se démarquer de ses voisins, afin de détecter des zones d'intérêts de l'image. (Alexe *et al.*, 2010) proposent d'utiliser des caractéristiques telles que le contraste de couleur, la densité des coins visuels et l'utilisation de cartes de saillance à plusieurs échelles afin de trouver le positionnement de régions potentiellement pertinentes. (Rybok, 2017) propose une idée similaire basée sur l'algorithme "**quaternion-based spectral saliency detection**", une approche qui fusionne les cartes de saillance avec les segments des formes de l'image afin de détecter les segments de saillance les plus distinctifs. Ces approches offrent un taux de reconnaissance d'objets élevé selon une approche non supervisée, mais nécessitent toutefois l'utilisation de divers algorithmes de segmentation et de classification additionnels pour la découpe des régions, ce qui peut amener à des redondances de systèmes. Ce problème est exacerbé dans le contexte de ce mémoire, puisqu'un robot assistant mobile nécessite des modules optimisés

afin de limiter la perte de ressources et de batterie pour son bon fonctionnement.

Une autre approche intéressante pour aborder cette catégorie de proposition de régions est l'algorithme **Selective Search** proposée par (Uijlings *et al.*, 2013). Cet algorithme, même si elle n'est pas la plus efficiente actuellement dans le domaine (Ren *et al.*, 2015), offre un bon ratio *précision/temps* sans nécessiter précedemment l'entrainement d'un modèle de caractéristiques utilisé par plusieurs de ses congénères (Zitnick et Dollár, 2014; Arbeláez *et al.*, 2014).
L'algorithme fonctionne de la manière suivante :

- L'image est segmentée en différentes sections selon un algorithme de découpe de régions par graphe, normalement effectué par l'algorithme de Felzenszwalb et Huttenlocher (Felzenszwalb et Huttenlocher, 2004).
- Pour chaque paire de régions possibles, on calcule le degré de similarité. Le degré de similarité du Selective Search se base sur la similarité des couleurs selon leurs histogrammes, la similarité des textures basée sur des caractéristiques SIFT (Liu *et al.*, 2010) et la similarité des emplacements des régions dans l'image.
- On extrait la paire de régions avec le plus grand degré de similarités, lesquelles sont fusionnées et ajoutées aux propositions en sortie de région. On enlève ensuite toute les paires de régions dont l'un des éléments est une des régions fusionnées.
- On calcule les degrés de similarités entre la nouvelle région et les autres propositions restantes qu'on rajoute à l'ensemble.
- L'algorithme redémarre l'étape 3 jusqu'à que l'ensemble de paires de régions proposées soit vide.

1.3.3 Classification

Dans le domaine de la classification d'objets pour une image, le critère essentiel est de choisir une méthode de classification multiclasse flexible aux différentes formes de caractéristiques extraites. Une majeure partie des techniques de classification du secteur de l'apprentissage machine est applicable afin d'effectuer cette catégorisation.

Le choix du classificateur dépend de la catégorie du problème :

— Y a-t-il une possibilité de trouver des pixels mixtes dans les images à classifier, c'est-à-dire les cas où une sous-région de l'image appartient, ou peut appartenir, à plusieurs catégories d'objets en même temps (Choodarathnakara *et al.*, 2012) ? Dans un cas où nos régions ne peuvent être découpées en zones distinctes, l'utilisation de classifieurs doux comme un système d'inférence flou peut apporter des résultats adéquats, mais moins précis que ceux des classificateurs durs.

— L'apprentissage doit-il être supervisé ou non supervisé ? L'utilisation d'un classificateur supervisé nécessite l'entrainement d'un modèle, ce qui peut s'avérer être une tâche difficile dans le domaine de la vision par ordinateur. En effet, un jeu d'entrainement optimal doit être assez grand et varié pour limiter l'effet néfaste des divers facteurs de variance discutés dans 1.2. Un bon module peut nécessiter des ensembles de données massives, ce qui n'est pas toujours réalisable lorsque la classification se porte sur des ensembles et sur des catégories de classification de petites tailles. Toutefois, l'utilisation de classificateurs non supervisés pour la classification d'objets d'une image est peu souvent incitée, puisque leur taux de précision et leur vitesse d'exécution sont significativement plus faibles que les techniques supervisées traditionnelles.

— Est-ce que les caractéristiques des objets sont facilement paramétrisables pour un classificateur ? Dans le cas où chaque objet est significativement distinct des autres classes, l'utilisation de classificateurs paramétriques tels que l'analyse discriminante linéaire ou les machines à vecteurs de support, offre un haut taux de précision à faible coût computationnel. Par contre, si l'ensemble de données est trop complexe pour être paramétré manuellement, l'utilisation de classificateurs non paramétriques comme un réseau de neurones est plus appropriée.

CHAPITRE II

APPRENTISSAGE PROFOND APPLIQUÉ À LA VISION PAR ORDINATEUR

Au dernier chapitre, les approches de base pour systèmes de reconnaissance de formes et de classification ont été discutées. Ces méthodes offrent un bon rapport entre la qualité des résultats et la rapidité des calculs effectués. Toutefois, la majorité de ces méthodes comportent la même lacune que les approches traditionnelles qu'on peut retrouver en apprentissage machine, c'est-à-dire l'extraction de caractéristiques moins riches en information, puisqu'il est difficile pour un chercheur de reconnaître quelles sont les meilleures caractéristiques. Cela peut causer une perte de qualité pour ce qui concerne la localisation et la classification (LeCun *et al.*, 2015). Ces limitations sont, qui plus est, exacerbées dans le domaine de la vision pour agent robotique mobile, puisque le système doit traiter des images qui changent continuellement d'angle et d'environnement, en plus de l'occlusion possible provoquée par la gigue de la caméra lors du déplacement du robot.

Aujourd'hui, une solution prend de plus en plus d'ampleur dans le domaine de vision par ordinateur, ainsi que dans plusieurs autres domaines de l'intelligence artificielle et de l'apprentissage machine, à savoir les **réseaux de neurones par apprentissage profond**. Au début des années 2010, les approches de l'apprentissage profond ont fait une ré-émergence majeure dans le milieu de la recherche,

ce qui a mené à leur utilisation dans divers projets nécessitant une classification fine, tels que la reconnaissance faciale (Le, 2013), la création d'IA pour divers jeux (Mnih *et al.*, 2013), le profilage des goûts musicaux pour générer des recommandations personnalisée (Wang et Wang, 2014) et bien plus encore.

Ces techniques sont aussi souvent utilisées dans le secteur de la vision par ordinateur et offrent des résultats comparables aux méthodes conventionnelles pour plusieurs secteurs, comme ceux de la reconnaissance d'objets, la reconnaissance de liens sémantiques entre différentes classes, etc.

Dans ce chapitre, on se penche sur l'utilisation des réseaux de neurones pour répondre à la problématique de la classification d'image. Une courte revue des réseaux de neurones est effectuée avant de poursuivre avec leur version adaptée au domaine de la vision par ordinateur, nommée les **Réseaux de neurones à convolution** (**CNN**). Finalement, on présente un état de l'art des dernières années sur les CNN et les approches qui permettent de les utiliser pour traiter en parallèle la problématique de détection et celle de classification de formes, un critère essentiel pour cette recherche en robotique.

2.1 Réseaux de neurones artificiels

L'utilisation de réseaux de neurones comme technique d'apprentissage a explosé ces quinze dernières années, grâce aux recherches sur l'application de l'apprentissage profond permettant son application à différents domaines (Bengio *et al.*, 2009). Ce type de technique a montré une importante capacité d'adaptation à différentes problématiques, où il parvient à offrir des résultats supérieurs à ce que qui est présenté dans l'état de l'art.

Un des problèmes majeurs des approches d'extractions des caractéristiques présentées au chapitre précédent, telles que l'utilisation de SIFT ou SURF, est que

les caractéristiques extraites présentent une limitation en ce qui a trait au type d'information percevable. L'algorithme SIFT, par exemple, une des bases en classification d'images, peut seulement fonctionner sur des images en nuances de gris et perd ainsi toutes les informations issues du spectre des couleurs. Une autre lacune de cet algorithme est que sa fonction d'extraction est invariante au jeu de données et ne peut s'adapter pour reconnaitre les caractéristiques uniques de l'ensemble. Les approches comme SIFT sont fixes et sont toujours exécutées de la même manière, peu importe si l'image d'origine contient 1 ou 50 objets différents. Cependant, cet inconvénient peut être résolu dans la mesure où le module de reconnaissance de formes et d'extraction de caractéristiques peut apprendre sa fonction d'extraction en fonction de l'ensemble classifiable. Pour ce faire, il n'est pas nécessaire de modéliser chaque caractéristique à la main, qui est une tâche infiniment complexe, mais plutôt d'utiliser un réseau de neurones.

La base des réseaux de neurones utilisés pour résoudre les problèmes de reconnaissance en imagerie est restée la même ces vingt dernières années : l'utilisation d'un réseau multicouche avec rétropropagation du gradient afin de mettre la fonction objectif de chaque neurone à jour (Kanellopoulos et Wilkinson, 1997).

La plupart des couches de réseau et leurs fonctions d'activation de base sont utilisables pour les réseaux de neurones en imagerie : la sigmoïde $sigmoïde(x) = \frac{1}{1+e^{-x}}$, l'unité de rectification linéaire $relu(x) = max(0, x)$, ou encore l'unité de rectification linéaire qui fuit, cette variante de la version traditionnelle conçue pour éviter le cas d'erreur où le poids du neurone devient négatif lors de l'entrainement (ce qui le rendrait inutilisable), par l'utilisation d'une constante θ située entre [0,1], qui est directement retournée si le neurone et son poids retournent une valeur négative (Maas *et al.*, 2013).

$$LeakyReLU(x) = \begin{cases} \theta x & \text{si } x < 0 \\ x & \text{si } x \geq 0 \end{cases}$$

L'utilisation d'un réseau de neurones multicouche traditionnel est toutefois peu intéressante dans le domaine de la classification en imagerie numérique, puisqu'un tel réseau détient une explosion combinatoire sur le nombre de paramètres, c'est-à-dire de neurones par couches nécessaires pour effectuer une classification d'objets avec un haut taux de précision. Par exemple, un réseau à deux couches cachées, ayant respectivement 4000 et 2000 neurones, pour un jeu d'entrainement d'images de taille 256×256 pixels selon 10 catégories d'objet possibles, nécessite un entrainement de $256 \times 4000 + 4000 \times 2000 + 2000 \times 10 + 10$, sans même compter les fonctions d'activation pour chaque neurone.

Ce nombre de paramètres est ensuite triplé si l'image comporte trois canaux de couleurs, puisque chacun d'entre eux, Rouge-Vert-Bleu, nécessite une paramétrisation unique. De plus, si on souhaite effectuer un entrainement efficient pour chacun de ces neurones, on a besoin d'un jeu de données immense et dont chaque élément est disjoint, pour assurer que le réseau soit entrainé pour chaque éventualité et pour éviter les cas possibles de **sous-apprentissage** et de **sur-apprentissage**. Ce dernier cas est particulièrement critique, puisque le taux d'invariance des paramètres vis-à-vis les critères de variance d'une image, tel que décrit au chapitre 1, est proportionnel à leurs apparitions dans les jeux de données.

Si le jeu d'entrainement ne comporte aucune image avec variations de taille, d'angle, de luminosité, etc. détectable sur les objets à classifier, alors les paramètres du réseau ne peuvent être entrainés pour détecter ces cas et, par conséquent, ne peuvent les classifier. Il arrive aussi souvent, lorsque le jeu de données est de petite taille et détient un haut taux de similarités sur son ensemble d'entraine-

ment, que le réseau effectue un sur-apprentissage et ne se paramétrise pas selon les objets classifiables, mais plutôt sur les données d'arrière-plan de l'entrainement. Lorsque cela arrive, le réseau peut seulement reconnaitre les images d'entrainement et ne donne pas de résultats significatifs pour n'importe quel ensemble de validations distinct.

2.2 Réseaux de neurones à convolution

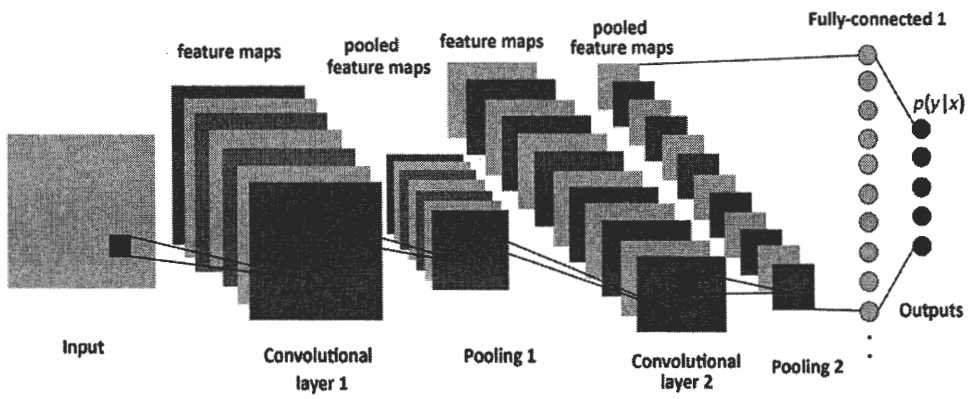


Figure 2.1: Représentation d'un réseau neuronal convolutif. Source : (Blanc-Durand, 2018)

Les réseaux de neurones à convolution, aussi appelés **réseaux neuronaux convolutifs**, sont une amélioration de l'approche traditionnelle des réseaux de neurones qui vise à limiter le nombre de paramètres et de couches nécessaires au fonctionnement du réseau tout en assurant une architecture riche ainsi qu'un taux de classification significatif.

Cette classe de modèle, basée sur le champ récepteur des neurones d'un cerveau, est apparue dans les années 1990 sous la forme d'un **time-delay neural network**. Toutefois, sa popularité est plus récente, basée sur les implémentations de réseau de neurones avec calculs effectués sur des cartes graphiques.

(L'entrainement d'un même réseau sur carte graphique comparé à un entrainement sur processeur seul montre pour certaines bibliothèques une amélioration de la vitesse d'entrainement pouvant être 50 à 60 fois plus rapide .[1])

Un exemple populaire dans le domaine est donné dans l'article *ImageNet Classification with Deep Convolutional Neural Networks* (Krizhevsky *et al.*, 2012) : des chercheurs de l'université de Toronto ont utilisé un réseau neuronal convolutif à 7 couches sur le jeu de données ImageNet de la compétition **ILSVRC-2010** et ont réussi à garder un taux d'erreur de classification plus bas d'au moins 8 % que celui des approches recensées par l'état de l'art du moment.

Tableau 2.1: Taux d'erreur pour 1 et 5 choix possibles de classe de CNN et des approches traditionnelles sur le jeu de données ILSVRC. Source : (Krizhevsky *et al.*, 2012)

Modèle	***Top-1***	***Top-5***
Réseau de neurones à codage parcimonieux	47.1%	28.2%
Caractéristiques SIFT + vecteur Fisher	45.7%	25.7%
CNN	37.5%	17.0%

En comparaison des réseaux multicouches traditionnels, la structure avec convolution utilise le fait que les différentes sous-zones de pixels d'une image contiennent des informations plus pertinentes à l'analyse pour leurs voisins immédiats que les zones distantes. En effet, il existe peu de situations pour une image où un pixel

1. `https://medium.com/@andriylazorenko/tensorflow-performance-test-cpu-vs-gpu-79fcd39170c`

n'ait pas de liens sémantiques avec ses voisins, puisqu'une caractéristique d'image n'est pas décrite par un pixel seul, mais plutôt par une région de pixels. (Même certaines approches conventionnelles, telle que SIFT, utilise ce concept pour détecter les maximums locaux entre un pixel et ses voisins lors du calcul de ses caractéristiques.)

De ce fait, il devient plus intéressant de formuler la structure des neurones du réseau de sorte qu'elle soit reliée à différentes régions de pixels ainsi qu'à leurs voisins immédiats, plutôt que d'implémenter un neurone par pixel. Il est ainsi possible de limiter le nombre de paramètres et le nombre de poids de chaque neurone au nombre de blocs de taille fixe pour chaque couche. La taille reste fixe pour chaque couche individuelle, mais diminue au fil du réseau afin de traiter les paramètres pour les caractéristiques de différentes tailles. Pour faire cette réduction, il est possible d'appliquer les convolutions de différents filtres sur l'image en entrée, afin de faire ressortir N sous images, emmagasinées dans un canal distinct où N est le nombre de filtres d'une couche. Cela nous permet de contrôler le nombre de canaux d'une image lors des étapes de l'entrainement et de la classification, en plus d'assurer un traitement minimal des trois canaux de couleurs qui étaient souvent ignorés des techniques présentées au chapitre précédent. Chaque sous-image peut être ensuite transférée aux fonctions d'activation traditionnelles du réseau pour la tâche de détection.

Un autre avantage de ce type de réseau est que les connexions ne sont pas de neurones à pixel comme dans une approche de réseau traditionnelle, mais plutôt de pixel à bloc, chaque bloc étant composé de neurones connectés à une région spécifique de l'image de taille fixe. Cette paramétrisation limite le nombre de connexions pour chaque neurone aux pixels de sa région, ce qui restreint significativement le nombre de connexions du réseau, en plus de permettre de réduire la paramétrisation des poids des neurones d'un bloc en utilisant seulement un

poids par bloc. De ce fait, il est possible de réduire le nombre de connexions d'un neurone dans une couche à seulement ceux qui sont intrinsèquement reliés à sa région. Ce type de réseau peut être encore amélioré par l'utilisation de plusieurs couches de convolution de filtres de taille unique afin de l'entrainer à reconnaitre les caractéristiques pour différentes résolutions.

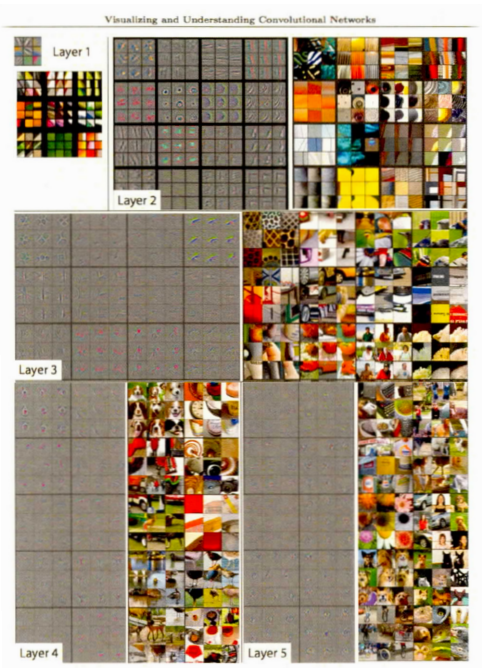


Figure 2.2: Visualisation des couches d'activation d'un réseau de neurones. Source : (Zeiler et Fergus, 2014)

2.3 État de l'art des classificateurs par apprentissage profond

Un des problèmes majeurs du CNN, hormis la quantité de ressources nécessaires à son entrainement, est que le modèle est seulement adapté aux couches de sortie des réseaux traditionnels, c'est-à-dire aux couches avec fonctions **Softmax** ou similaires. Cette limitation fait que le réseau est adapté seulement pour fonctionner selon le principe *recevoir une image en entrée, la transférer directement dans le réseau et recevoir une donnée en sortie.* Or il est rare pour un système de classi-

Figure 2.3: Zones d'activation de la 5e couche du réseau de neurones du mémoire sur la même entrée que la figure 1.3.

fication en imagerie, surtout pour la conception d'agents robotiques, que l'image en entrée soit traitée parfaitement avec cette contrainte. Normalement, les images captées par un robot assistant présentent un taux variable d'objets et/ou de personnes à reconnaitre. Même dans certains cas où l'agent robotique n'a qu'un seul objet à détecter, il est possible que cet objet soit confondu avec l'arrière-plan de l'image si l'arrière-plan détient des caractéristiques proches que le réseau est entraîné à détecter, ce qui peut générer de faux positifs. Pour effectuer les phases de détection et de classifications d'objets en parallèle, il est nécessaire pour le système d'avoir accès à des **propositions de régions**, aussi appelées **cadres de limitation** (**Bounding box** en anglais), c'est-à-dire des régions qui peuvent être automatiquement proposées par le système comme emplacement possible d'un objet classifiable. On souhaite aussi effectuer **une régression linéaire des propositions de régions**, afin d'éliminer l'aire inutile dans un cadre, c'est-à-dire d'entourer l'objet le plus proche possible afin d'éviter de laisser du vide autour.

Figure 2.4: Représentations de différents cadres de limitation de zones avec un pichet. L'utilisation d'une régression linéaire permet d'optimiser le positionnement des coins du cadre pour contenir seulement la région d'intérêt (**ROI**) du pichet.

Pour contourner cette problématique, on peut utiliser une approche par réseau neuronal convolutif avec un module de génération de régions. Un exemple de ce type d'approche est la famille des algorithmes **R-CNN**.

Des chercheurs de Berkeley ont publié un article, *Rich feature hierarchies for accurate object detection and semantic segmentation* (Girshick *et al.*, 2014) où ils proposent d'utiliser l'algorithme de recherche sélective décrit au chapitre précédant pour générer des propositions de régions possibles dans une image. Ces régions sont ensuite transformées en vecteurs de caractéristiques, afin d'être introduites dans un réseau CNN à cinq couches de convolution et deux couches **entièrement connectées**, des couches ou chaque neurone est lié aux actions de la couche précédente. Ce réseau est paramétré selon les activations sur chaque couche lors de l'entrainement afin de l'optimiser, avant de finalement extraire les caractéristiques de sa dernière couche pour entrainer un **SVM** sur chaque classe classifiable par la

technique d'***apprentissage par transfert***. Cette approche offre de bons résultats en comparaison d'autres méthodes, mais ce générateur de régions est difficile à intégrer au CNN car il nécessite d'envoyer chacune des propositions de régions au réseau, incluant celles présentant des taux de similarité élevés, ce qui cause de la redondance de calculs et un goulot d'étranglement majeur sur la vitesse de traitement totale. Même si l'algorithme de génération de régions est remplacé par l'une des alternatives décrites au chapitre précédent, la redondance des calculs va toujours provoquer une perte en efficience.

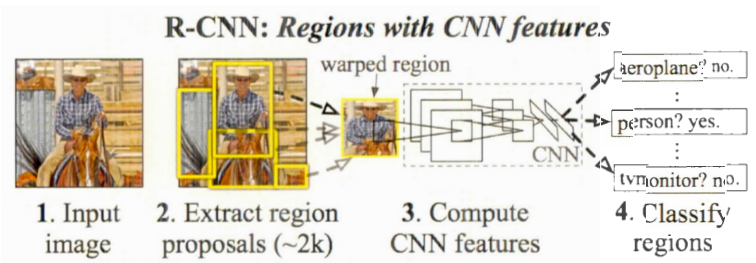


Figure 2.5: Représentation de l'architecture d'un réseau neuronal convolutif avec régions. Source : (Girshick *et al.*, 2014)

Le chercheur principal de l'article original sur les R-CNN, Ross Girshick, a publié un an plus tard une évolution de son approche qu'il dénomme **Fast R-CNN** (Girshick, 2015). Plutôt que d'effectuer une détection pour chacune des propositions de région, l'algorithme prend en entrée l'image à classifier et un jeu de propositions de régions préparé à l'avance. Le réseau calcule les caractéristiques de l'image par plusieurs couches de convolution et de réduction, avant de les passer à une couche, appelée *ROI pooling*, qui réduit le jeu des caractéristiques à celles qui existent seulement dans les régions proposées.

Ce même auteur a publié une dernière amélioration de son approche la même année, afin de traiter la problématique de la génération de régions, un problème mis de côté par la version précédente. Plutôt que d'émettre la supposition selon laquelle le réseau de neurones reçoit toujours un vecteur de régions probables lors de l'étape de la classification, l'article *Faster r-cnn : Towards real-time object detection with region proposal networks* (Ren *et al.*, 2015) propose l'utilisation d'un réseau neuronal convolutif intermédiaire, nommé **réseau de proposition de région (RPN)**, qui apprend au réseau de classification la manière optimale de générer des propositions de régions selon ses classes d'objets. Premièrement, Girshick utilise une architecture de réseaux de neurones composés de plusieurs couches de convolution pour extraire des vecteurs de caractéristiques. Ceux-ci sont ensuite transférés à une fonction de fenêtre mobile de taille fixe, qui a pour but de proposer des régions de différentes tailles autour de plusieurs points d'ancrage de dimensions variables. Ces points sont paramétrés manuellement par les utilisateurs, afin de pouvoir extraire des régions de différentes hauteurs et largeurs pour différentes positions possibles dans les images. Ces propositions sont ensuite transférées à une couche de régression de cadre et à une couche de classification qui les classifie selon la forme **Détient Objet-Aucun objet**, afin de permettre au réseau d'offrir à sa sortie un vecteur de régions proposées ainsi que le niveau de confiance qu'un objet est réellement situé dans cette région.

Lorsque ce réseau est entrainé, il peut être utilisé pour entrainer à son tour un réseau de classification R-CNN en lui offrant des vecteurs de régions probables. Ce second réseau présente une architecture semi-partagée à celle du RPN, grâce à la réutilisation des mêmes couches de convolution initiales. Ce partage permet ainsi, après avoir fini d'entrainer le classificateur d'objets, d'effectuer deux phases de transfert de connaissances en fixant la paramétrisation des couches existantes dans les deux réseaux et en effectuant une optimisation paramétrique seulement sur les couches uniques de chacun.

Plusieurs améliorations existent pour cette famille d'algorithmes, telles que l'intégration de système de segmentation de formes afin d'offrir des régions plus fines pour les objets classifiés (He *et al.*, 2017; Chen *et al.*, 2017) ou encore la modification des fonctions d'activation du réseau afin de déterminer quelle classe apparaît le plus souvent au premier plan (Lin *et al.*, 2018). Malheureusement, ces approches n'empêchent pas la surutilisation des ressources nécessaires à l'entrainement et au déploiement de deux réseaux, ce qui les rend peu intéressants pour des systèmes à ressources limitées tels ceux d'un robot assistant.

Une autre amélioration apparue ces dernières années est l'utilisation de **blocs de résidus** dans l'architecture d'un réseau (He *et al.*, 2016). Ces blocs sont placés après des couches de convolution et servent d'intermédiaires pour le transfert de gradients entre différentes couches du réseau. Plutôt que d'effectuer la propagation du gradient couche par couche, le bloc de résidus fixe une couche précédente comme valeur de raccourci à la fonction d'activation actuelle. Cela permet au gradient d'être transféré à une couche plus basse du réseau directement et permet aussi de fusionner le résultat de plusieurs couches. Bien utilisé, les blocs de résidus permettent à un réseau d'effectuer plusieurs classifications en parallèle sur différentes tailles de filtres, un avantage lors d'une classification de catégories d'objets de tailles variables.

2.3.1 YOLO : *You only look once*

Conçu et continuellement amélioré par ses auteurs Joseph Redmon et Ali Farhadi de l'université de Washinton (Redmon *et al.*, 2016; Redmon et Farhadi, 2017; Redmon et Farhadi, 2018), ***You only look once*** (YOLO) est un algorithme de réseau neuronal convolutif avec sélection de régions, similaire à ceux présentés à la section précédente. À la base, YOLO utilise un modèle de réseau basé sur *Googlenet* (Szegedy *et al.*, 2015), qu'il nomme *Darknet*, avec modifications sur les

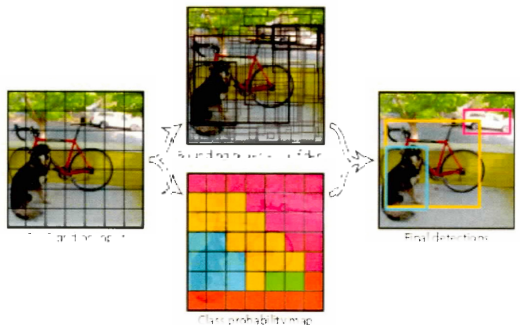


Figure 2.6: Structure YOLOv1. Source : (Redmon *et al.*, 2016)

couches en entrée afin que s'effectuent les phases de détection et de classification à l'intérieur même du réseau, sans nécessiter l'utilisation d'un second réseau.

Plutôt que d'entrainer son modèle et ses fonctions d'activation sur des régions spécifiques d'une image, YOLO les entraine sur l'image d'entrée dans son entièreté avant d'effectuer une phase de localisation de régions probables. Cette approche permet d'assurer que les vecteurs de caractéristiques pour chaque couche du réseau gardent leurs données de contexte spatial pertinentes, afin d'éviter des faux négatifs si les régions proposées ne détiennent pas la totalité de l'image.

Pour ce faire, YOLO divise l'image en entrée en cellules de taille 32×32. Pour chaque cellule, YOLO traite 5 cadres d'ancrages possibles, aussi appelés des *clusters de dimensions*. Ils sont calculés à l'avance par son utilisateur avec l'algorithme **clustering de k-moyennes** avec une métrique de distance ***d'intersection sur union*** (**IOU**). Ces cadres sont utilisés lors de l'entrainement et de la classification pour calculer, pour chaque cellule, le niveau de confiance qu'il détient d'un objet, le taux d'erreur entre la taille de l'objet trouvé comparée à celle modélisée par le réseau et le taux de confiance que l'objet est l'unes des classes du connues par le réseau. Grâce à ces données, le réseau est capable, par réduction de caractéristiques dans ses dernières couches de convolution, de transmettre une grille qui

divise l'image en zones de taille 32×32, où chaque zone contient les cadres de limitations avec les taux de confiance de contenir un objet le plus élevé, ainsi que les niveaux de confiance que l'objet appartiennent à une des k classes connues par le système. Cette approche permet d'offrir un système tout-en-un, capable de faire la localisation et la classification sur des caractéristiques riches, sans toutefois causer un goulot d'étranglement pour la phase de détection d'objets.

CHAPITRE III

RECONNAISSANCE D'ACTIVITÉS DE TYPE HUMAIN-OBJET POUR UN AGENT ROBOTIQUE

Les chapitres précédents ont majoritairement porté sur les systèmes de vision numérique utilisés en intelligence artificielle, mais n'ont pas beaucoup discuté de la raison de leur utilité dans cette recherche ; c'est-à-dire de leur utilisation pour la reconnaissance d'activités dans le cas d'un robot assistant cognitif. Ce type d'assistants robotiques, plutôt que d'essayer de créer une plateforme pour effectuer des interactions physiques avec une personne dans le but d'offrir une aide physique à la réalisation de ses activités de vies, veut analyser les actions de son utilisateur, ses habitudes et ses méthodes d'action, et ce, afin de comprendre ses besoins. Ces informations lui permettent d'offrir une assistance cognitive pour effectuer ses activités de la vie quotidienne.

Pour donner un exemple, un agent robotique cognitif ne cherche pas à aider son utilisateur en allant *chercher ses pantoufles*, mais plutôt en comprenant que la personne effectue la tâche *chercher pantoufles* et utilise cette information pour *avertir la personne de leur position.* Afin de fonctionner, l'agent robotique a besoin d'un moyen de visualiser son environnement et des méthodes pour détecter les différentes activités effectuées par une personne dans son quotidien.

Dans ce chapitre, on penche notre analyse sur les approches actuelles de création d'agents robotiques assistants. Une courte revue de l'art sur les assistants cognitifs est effectuée. On continue ensuite par une comparaison entre les approches créées pour des agents fixes et amovibles, c'est-à-dire fixés sur place ou capables de se déplacer et des approches conçues pour des robots assistants afin de comprendre leurs avantages. On effectue ensuite un état d'art rapide des méthodes courantes de reconnaissance d'activités des agents robotiques et leurs limitations. Une emphase est mise après sur le concept de **l'affordance** et son utilité dans le mémoire. Finalement, le chapitre se termine sur les motivations et les objectifs du mémoire.

3.1 Assistant cognitif

Dans la littérature, une démarcation est faite entre les différentes formes d'assistants robotiques selon leurs buts primaires. Selon (Feil-Seifer et Mataric, 2005), un assistant robotique cognitif fait partie de la famille des robots assistants sociaux. Comparativement aux robots assistants physiques qui visent à offrir une aide physique à leurs utilisateurs, par exemple des actions tel que *aider à se lever*, *prendre un livre d'une armoire et nous le passer* ou même *préparer et servir la nourriture*, un robot assistant social vise à comprendre le besoin actuel de son utilisateur et à lui offrir une interaction appropriée.

Selon (Pollack *et al.*, 2002), un robot assistant cognitif ne sert pas à remplacer les interactions humain-humain d'une personne, mais plutôt à les enrichir. Ce type d'assistant offre une aide cognitive à son utilisateur afin de limiter les pertes cognitives subies, que ce soit celles dues à l'âge, à la maladie ou à un traumatisme. Cette aide doit permettre à une personne en perte cognitive de pouvoir effectuer ses activités de vie quotidienne tout en gardant un niveau d'autonomie élevé.

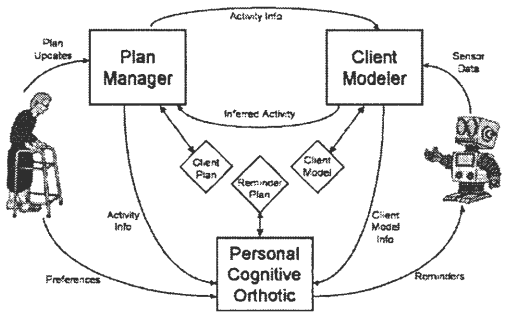


Figure 3.1: Représentation d'une architecture pour robot assistant cognitif. Source : (Pollack *et al.*, 2003)

Plusieurs systèmes d'assistants cognitifs sont recensés par la littérature. On retrouve par exemple le système **Autominder** (Pollack *et al.*, 2003). Autominder est un module intégrable à un assistant robotique qui sert à modéliser les activités de l'utilisateur. Ceci permet au robot assistant de reconnaitre ses habitudes afin de lui offrir des rappels aux moments appropriés. Ce système permet de donner une aide cognitive grâce à ses rappels à l'utilisateur, mais il est limité sur les activités modélisables par le système, ainsi que par son système de collecte de données qui nécessite des ajouts manuels externes pour reconnaitre des divergences dans l'exécution des tâches.

D'autres systèmes, tels que **Dem@Home** de l'institut des technologies de l'information (ITI) du centre pour la recherche et la technologie à Hellas en Grèce (CERTH) (Andreadis *et al.*, 2016), utilisent un système de capteurs placés sur l'utilisateur et son environnement afin de comprendre les activités d'un sujet selon ses observations et son ontologie. Ses observations sont aussi transmises à des experts en santé afin qu'ils les utilisent pour comprendre les besoins spécifiques de l'utilisateur.

Ces types d'approches, notamment celles implémentées dans des maisons intelligentes, permettent une modélisation fine des activités d'une personne dans le but de comprendre ses besoins, mais sont intrusifs dans la vie de l'utilisateur du fait du nombre élevé de capteurs sensoriels nécessaire pour leurs fonctionnements. Dans le cadre de ce mémoire, on cherche à offrir un système de reconnaissance d'activités simples qui peut être intégré dans une maison d'une personne âgée sans nécessiter de travaux majeurs pour son installation.

3.2 État de l'art pour robot assistant cognitif

À sa base, un robot assistant fonctionne selon un des principes de bases de l'intelligence artificielle, c'est-à-dire le concept de l'agent intelligent autonome. Selon (Russell et Norvig, 2016), un agent intelligent autonome est un agent autonome qui observe son environnement grâce à divers capteurs. Il interprète ensuite cette information et agit en conséquence afin d'atteindre ses objectifs, par le biais d'activités. Pour un robot assistant cognitif, le système fonctionne dans un environnement *stochastique* et *partiellement observable*, c'est-à-dire qu'il inclut *une notion de hasard* et que l'environnement ne peut être entièrement modélisé et nécessite donc *une représentation abstraite*. Un robot assistant s'apparente aussi au modèle d'*agent intelligent par modèle et par utilité*, c'est-à-dire que l'agent détient une représentation abstraite de l'environnement qu'il met constamment à jour par ses observations et une fonction d'utilité qui lui permet d'analyser le ratio gain/perte de ses actions afin de maximiser le gain de ses choix.

Un des principaux choix de conception à faire pour un robot assistant et important à traiter pour les besoins de ce mémoire est son lien avec son environnement. Un robot fixe, intégré à une maison permet l'utilisation d'un grand nombre de capteurs pour collecter des données et ainsi, une modélisation plus fine des activités d'une personne. Un robot amovible, tel qu'une plateforme à roulette, permet un niveau

de contrôle additionnel sur les positionnements des capteurs et une interaction plus directe avec un utilisateur. Il est aussi moins intrusif qu'une approche fixe, puisque le système peut se limiter à la plateforme mobile, sans nécessiter l'apport de changements de l'espace de vie de la personne âgée pour intégrer les différentes composantes du robot.

3.2.1 Agents fixes

Les approches fixes dans la littérature sont souvent implémentées par l'utilisation de **maisons intelligentes**. L'utilisation de l'espace de vie comme plateforme permet d'acquérir un plus grand niveau d'observation par l'installation de capteurs de pression ou de mouvement à des emplacements stratégiques dans une maison (porte d'entrée, porte du frigo, meuble dans la cuisine, etc.). Ces observations peuvent être réalisées sous différents formats (audio, vidéo, poids, etc.) et permettent d'enrichir certains modèles d'activités spécifiques.

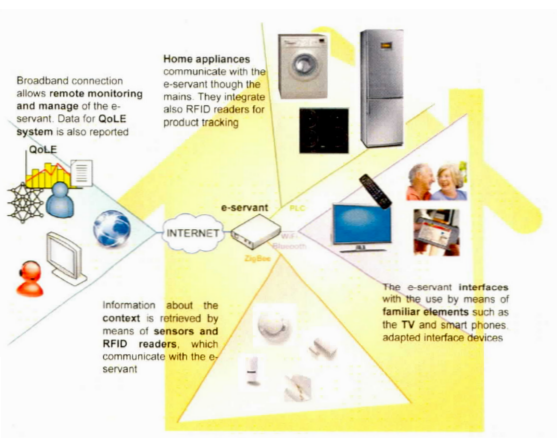


Figure 3.2: Architecture pour une cuisine intelligente. Le système utilise un couplage de divers capteurs et de lecteurs RFID sur chaque élément observable de la cuisine afin de déterminer avec quel objet l'utilisateur est en pleine interaction. Source : (Blasco *et al.*, 2014)

Dans leur article *Activity recognition on sensor streaming data* (Krishnan et Cook, 2014), Krishnan propose d'utiliser des capteurs positionnés sur les objets situés dans une maison, tels qu'une armoire, un four, etc., afin de modéliser leurs états. Ces observations, sous forme de liens états-objets avec les états *OUVERT* et *FERMÉ*, sont découpées en vecteur de caractéristiques d'activités distincts par section de temps, $[t_k..t_i]$ où k est le contexte temporel du début de l'activité selon les modèles de connaissance. Ces caractéristiques d'activités sont ensuite fusionnées selon leur degré de similitude, tel que la distance des différents capteurs d'activités durant ce segment temporel. Les caractéristiques sont finalement majorées par les probabilités de classification de l'analyse précédente, $S_i - 1$, et sont ensuite transmises à un SVM pour classifier l'activité en cours. Le système prend aussi en compte l'idée de *l'activité vide*, c'est-à-dire que les caractéristiques temporelles classifiées n'appartiennent à aucune activité. Ce choix de conception permet de modéliser et de traiter les cas d'apparition d'activités fantômes, c'est-à-dire de collecte de données même si aucune activité n'est en cours.

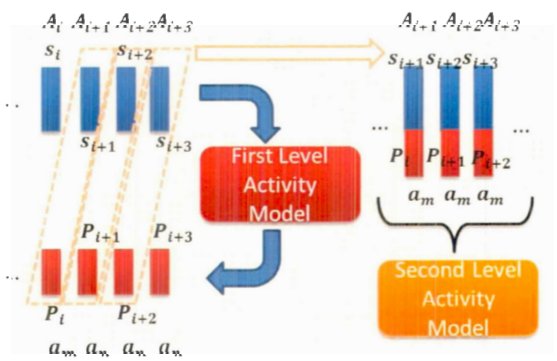


Figure 3.3: Classification d'activités par majoration des caractéristiques avec les valeurs de la classification précédente. Source : (Andreadis *et al.*, 2016)

Comme discuté précédemment, ces approches offrent une bonne qualité d'observations, mais nécessitent l'installation d'un nombre additionnel de capteurs à l'intérieur du domicile de l'individu recevant l'aide cognitive. Or, le projet décrit dans

ce mémoire a pour but d'offrir une solution d'assistant robotique cognitif à faible coût d'installation et qui doit limiter ses méthodes de collecte des données aux capteurs déjà installés sur la plateforme robotique, c'est-à-dire une caméra et un micro. Cette limitation rend alors l'utilisation d'assistants fixes inintéressants.

3.2.2 Agents amovibles

Les approches amovibles dans la littérature sont souvent implémentées par l'utilisation de **plateforme robotique mobile**. Ces approches sont plus limitées que leur congénère fixe, puisque leurs systèmes d'observation de l'environnement peuvent être limités aux capteurs placés sur la plateforme robotique. L'avantage de ce type d'approche réside dans le fait que le système est doté d'une plateforme amovible, ce qui lui permet de se déplacer dans l'environnement et ainsi de changer son jeu d'observation selon ses besoins. Ce besoin de se déplacer pour retrouver son propriétaire peut toutefois causer des situations où le robot assistant va manquer des observations critiques pour reconnaitre une activité puisqu'il n'est pas situé dans la même pièce, ou puisqu'il puisse être incorrectement placé pour visualiser l'action, etc. Cette approche détient tout de même l'avantage de pouvoir positionner ses capteurs à des endroits stratégiques dans l'environnement, grâce à un modèle des positions temporelles de la personne âgée qui effectue ses activités quotidiennes.

L'un des premiers systèmes d'assistance robotique cognitive sur plateforme mobile est le projet **Pearl**. Conçu par les universités du Michigan, Pittsburgh et Carnegie-Mellon dans le cadre du projet Nursebot. Pearl est une implémentation sur plateforme mobile du système Autominder, décrit dans la section 3.1, ainsi qu'une architecture haut niveau pour la prise de décision des actions d'aide cognitive, de communication et de mouvements, implémentés à l'aide d'une version hiérarchique d'un processus de décision markovien partiellement observable.

Figure 3.4: Le système Pearl déployé dans une maison de retraite. Source : (Pollack *et al.*, 2002)

Outre la plateforme Pearl décrite précédemment, on peut retrouver d'autres solutions d'assistants amovibles dans la littérature. Le système RAMCIP, théorisé par le centre de recherche en technologies Hellas de l'institut des technologies de l'information en Grèce (Kostavelis *et al.*, 2015), est pensé pour offrir à un robot assistant, en plus des systèmes usuels de modélisation, d'environnement et de planification des activités d'un utilisateur qu'il peut exécuter et un module de planification de tâches physiques afin d'offrir une aide à l'exécution d'activités physiques souvent oubliées et/ou ratées par l'utilisateur.

Une des difficultés de conception pour ce type de robot est la gestion des ressources. Hors les cas où la plateforme communique avec un second système fixe, un robot assistant amovible nécessite la gestion de toutes ses composantes physiques et informatiques selon les ressources de la plateforme seule. Les différents modules du système, tels que le déplacement, la reconnaissance d'activités, la reconnaissance de plan, le système de recharge, le système de communication avec l'utilisateur, etc. doivent être conçus pour fonctionner en parallèle sur un ensemble de ressources partagées. Toutefois, ce type de système d'assistance s'intègre mieux

dans un environnement existant puisque son intégration peut être limitée à l'addition de la plateforme à l'intérieur de son nouveau foyer. Le design de la plateforme permet aussi une intégration sociale avec son utilisateur, puisque sa forme physique et sa capacité d'interaction physique avec son propriétaire permettent de tisser un lien émotionnel entre les deux (Hutson *et al.*, 2011; Gross *et al.*, 2015).

3.3 État de l'art de la reconnaissance d'activité

Dans la littérature, on mélange souvent les termes «reconnaissance d'activité» avec les termes «reconnaissance de plan». Ce mélange est dû au fait que le terme *activité* peut désigner des activités de bas niveau, ce qu'on appelle aussi des actions élémentaires, et des activités de haut niveau, ce que l'on appelle aussi les plans et ou les buts d'une personne. Par exemple, l'action *prendre une assiette* est une activité élémentaire et est donc du domaine de la reconnaissance d'activité, mais elle n'est pas du domaine de la reconnaissance de plan, puisqu'elle ne permet pas de comprendre la finalité qu'on veut atteindre par cette action. Pour le cas spécifique de la reconnaissance d'activités par vision numérique, le terme «reconnaissance d'activité» désigne la détection et la reconnaissance des actions de bas niveau seulement. Toute analyse de haut niveau est désignée selon le terme «reconnaissance de plan». On doit tout de même prendre en compte que la détection d'activité est souvent utilisée comme méthode de collecte d'observations pour la reconnaissance de plan, un concept utilisé dans ce mémoire dans la section 4.5.

Un robot assistant cognitif nécessite la capacité de reconnaitre les activités et les plans d'une personne. Puisque son aide est seulement cognitive, il est essentiel de pouvoir reconnaitre rapidement les actions que la personne âgée peut faire, afin de déterminer quels sont les cas possibles d'erreurs auxquels on pourrait remédier en lui apportant l'aide nécessaire. En général, un module de reconnaissance d'activité pour un robot amovible est composé au moins d'un module de reconnaissance

d'être humain, de reconnaissances de caractéristiques spécifiques à une activité et d'une classificatrice pour inférer l'activité exacte observée.

3.3.1 Détection humaine

Une des techniques de détection du corps humain que l'on peut trouver dans la littérature est la méthode de suivi du squelette. Cette technique consiste en la détection et la modélisation des différents joints du corps humain, afin de bâtir une représentation simplifiée de son corps, que l'on décrit comme un squelette. Originellement, cette méthode était peu utilisée due à la complexité de la tâche de détecter chaque joint majeur du corps humain, mais elle a connu une ré-émergence majeure dans les dernières années après l'apparition de capteurs à faible prix pouvant offrir ce suivi, tel que le Microsoft Kinect.

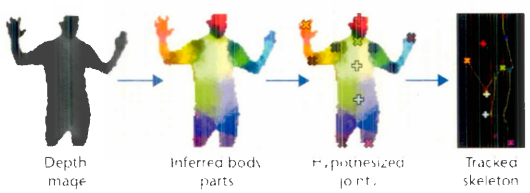


Figure 3.5: Calcul des joints du squelette humain. Source : (Zhang, 2012)

Certains chercheurs tels que (Liciotti *et al.*, 2017) se sont toutefois penchés sur la question de savoir si le suivi complet du corps humain était nécessaire ou si une généralisation avec des reconnaissances de certains membres spécifiques pouvait donner des résultats comparables. Dans leur article, *Hmm-based activity recognition with a ceiling rgb-d camera* (Liciotti *et al.*, 2017), les chercheurs Liciotti, Daniele et Duckett ont placé une caméra RGB-D sur un plafond fixant un utilisateur en train d'effectuer des activités de la vie quotidienne dans une cuisine. Leur système effectue une segmentation de l'arrière-plan basé sur une fenêtre temporelle pour détecter la forme appartenant à l'être humain en cours de déplacement.

La zone résultante est ensuite analysée afin de trouver le minima local de profondeur, ce qui représente la tête de l'humain observé et l'enlève de l'image avant de recalculer le minima local de nouveau, afin de déceler ce qu'ils considèrent être le second objet le plus haut placé : les mains.

Le nuage de points 3D des zones de mains et de tête extraits est ensuite envoyé dans un modèle de Markov caché comme observation afin de classifier l'action vraisemblable en cours selon la formule

$$x_j = argmax_i PHMM_i(X_{1:n} \in seq_n(s)|o_{1:n})$$

où PHMM est un modèle markovien partiellement caché et $X_{1:n}$ sont les actions possibles entre 1 et n, pour une séquence d'observation $\{s_0, s_1, .., s_n\}$.

Cette approche permet d'atteindre des taux de classification de 79 % pour des activités simples telles qu' *interagir avec la cafetière*, *interagir avec une théière* ou *interagir avec un frigo*. Une limitation est toutefois décelée dans sa méthode d'acquisition des données, puisque la structure d'extraction des mains n'est pas applicable dans un contexte général. Tout de même, cette d'approche démontre que plusieurs activités, notamment celle nécessitant une interaction avec des mains, peut être effectuées par une reconnaissance de zones spécifiques du corps humain.

3.3.2 Modélisation et classification des activités

Dans la littérature, la modélisation des caractéristiques d'une activité est souvent effectuée par diverses méthodes d'extraction d'images, telles que décrites dans le chapitre 1, qui sont enrichies pour faire ressortir les informations pertinentes propres aux activités avant d'être envoyées à un classificateur. Les approches sont tellement nombreuses et se basent souvent sur les mêmes algorithmes. Ainsi, il est plus intéressant de se pencher sur celles de certains chercheurs spécifiques.

Dans leur article *Spatio-temporal depth cuboid similarity feature for activity recognition using depth camera* (Xia et Aggarwal, 2013), les chercheurs Lu Xia et J.K. Aggarwal proposent de classifier les différentes actions d'un utilisateur par l'utilisation de caractéristiques 3D pavées droit centré autour de points d'intérêts spatio-temporels. Ces caractéristiques sont entrainées sur des vidéos de profondeur. Leur système utilise un filtre gaussien suivi d'un filtre temporel afin de filtrer les différentes formes spatiales de la vidéo. Ceci permet de maximiser les valeurs de pixels qui subissent un mouvement de flux optique. Ces données sont ensuite envoyées à une fonction de correction qui filtre les pixels subissant du bruit. Ce dernier est dû à l'apparition d'artéfacts lors de la numérisation des données reçues du capteur. Les caractéristiques spatio-temporelles sont ensuite extraites et un pavé droit est ensuite créé pour les entourer afin de pouvoir extraire des zones spécifiques des vidéos de profondeur traitées. Finalement, des histogrammes d'intensité de profondeur sont calculés pour chaque pavé droit, ce qui donne des vecteurs de caractéristiques qui peuvent être utilisés par un classificateur K-moyennes. La méthode réussit à atteindre un taux de classification d'activité de 88.2 % sur des activités simples telles que *se lever, s'asseoir, mouvement de main de va-et-vient*, mais aucun test n'a été effectué pour des actions complexes qui nécessitent une interaction avec l'environnement.

D'autres approches du domaine cherchent à utiliser les activités comme sources d'observations pour des plans de haut niveau. En 2011, les chercheurs de l'université Cornell ont utilisé un modèle de Markov caché hiérarchique afin de modéliser des plans comme suite de sous-activités (Sung *et al.*, 2011). Par exemple, l'activité *boire de l'eau* peut être reconnu par les caractéristiques des sous-activités qui se suivent de *prendre verre*, *remplir verre avec de l'eau* et *boire un liquide d'un verre*.

Leur approche permet de classifier différentes activités par l'utilisation de caractéristiques corporelles d'une personne, telle que l'estimation de la pose prise selon les données du squelette et l'angle des mains. Ils obtiennent un taux de classification d'activité pouvant atteindre jusqu'à 92 % et qui permet de faire la démarcation entre plusieurs plans avec une exécution similaire, mais avec une terminaison unique, tel que représenté par un exemple dans le tableau 3.1. Leur approche est toutefois limitée par leur modélisation d'une activité restreinte uniquement au corps humain.

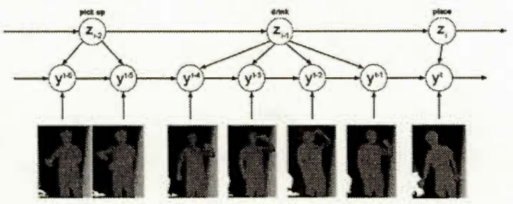


Figure 3.6: Représentation de modèle de Markov caché pour le lien hiérarchique entre l'activité *boire de l'eau* et ses sous-activités et leurs liens hiérarchiques. Source : (Sung *et al.*, 2011)

Tableau 3.1: Activités *Préparer une omelette* et *Préparer une omelette au jambon*

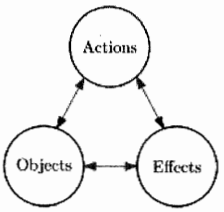
Activité	**Cuire omelette**	**Cuire omelette au jambon**
Étape-1	Placer poêle sur plaque chauffante	Placer poêle sur plaque chauffante
Étape-2	Démarrer plaque	Démarrer plaque
Étape-3	Briser œuf dans poêle	Briser œuf dans poêle
Étape-4	Attendre que l'œuf cuit	Attendre que l'œuf cuit
Étape-5	Mettre omelette dans assiette	Ajouter jambon dans l'omelette et continuer a cuire
Étape-6	Fin	Mettre omelette dans assiette
Étape-7		Fin

Le tableau 3.1 représente la série d'activités pour exécuter les plans *Préparer une omelette* et *Préparer une omelette au jambon*. La suite des activités des deux plans est similaire jusqu'aux dernières étapes, où l'omelette au jambon diffère par des étapes additionnelles nécessaires.

3.3.3 L'affordance

Terme créé par le psychologue James Gibson dans son livre *The Ecological Approach to Visual Perception* (Gibson, 1979), l'affordance était originellement pensée pour décrire ce que l'environnement d'un agent, qui dans le cas du livre de Gibson était un animal, peut lui offrir afin de montrer leur complémentarité. Gibson suit cette description avec un exemple basé sur la relation agent-sol terrestre «Si une surface terrestre est presque horizontale (plutôt qu'inclinée), presque plate (plutôt que convexe ou concave), est assez prolongée et est rigide, alors la surface offre une relation de support.». La description continue ensuite pour décrire que ce support permet de modéliser différentes interactions entre le sol et l'agent, telles que *se tenir debout*, *marcher*, *courir*, et ne permet pas d'autres supports possibles pour d'autres objets tels que *couler*. Le terme a été repris plus tard par le chercheur Donald Norman, qui l'a généralisé pour décrire les choix possibles d'actions qu'un objet offre à son agent.

Le concept de l'affordance est utilisé en psychologie pour décrire l'hypothèse que le cerveau humain modélise des connaissances sur les différentes manières avec lesquelles nous pouvons interagir avec les objets de notre environnement. Cette hypothèse a été testée à maintes reprises dans la communauté scientifique, par exemple par (Grezes et Decety, 2002) qui ont analysé les zones d'action du cerveau humain lors de la reconnaissance d'activités simples afin de déterminer que la voie centrale, reliée à l'identification des objets, était peu utilisée, ce qui suggère que l'activité humain-objet est modélisé indépendamment de l'objet.



Inputs	Outputs	Function
(O, A)	E	Effect prediction
(O, E)	A	Action recognition/planning
(A, E)	O	Object recognition/selection

Figure 3.7: Représentation des liens d'affordance entre les objets, les actions et leurs effets. Source : (Lopes *et al.*, 2007)

Le concept a finalement été adapté en informatique pour enrichir le modèle d'activités dans les contextes humains-objets. Les affordances permettent de modéliser les activités de bas niveau comme des interactions entre l'agent suivi et son environnement. Lors de la phase de reconnaissance d'activité, on peut utiliser ce principe pour décrire les interactions entre une personne et les différentes zones d'un objet comme caractéristiques spécifique à une action donnée. Prenons comme exemple la figure 3.8. L'interaction entre une personne et les différentes zones colorées permettent de décrire deux activités différentes telles que *boire* si la personne tient la tasse par l'anse, ou *remplir* si la tasse est tenue par la soucoupe.

En 2013, le chercheur Hema Swetha Koppula a publié plusieurs articles où il utilise l'affordance pour modéliser un objet et les formes d'une main qui interagit avec afin de représenter leurs états à travers le temps (Koppula et Saxena, 2013b; Koppula et Saxena, 2013a; Koppula *et al.*, 2013). Son approche utilise l'affordance pour modéliser des états de champ aléatoire conditionnel pour modéliser des plans composés de plusieurs activités, sous format main-objets, selon une démarcation temporelle. Son approche montre de hauts taux d'anticipation d'activités et d'affordance future battant l'état de l'art et un taux de reconnais-

Figure 3.8: Tasse avec ses différentes zones d'affordances.

sance de plan d'environ 85 %. Toutefois, l'approche a de la difficulté face à la problématique de la reconnaissance d'objets, moyennant un taux de précision de 56 % sur un modèle de 6 classes d'objets. Un facteur intéressant qui ressort des divers articles de ce chercheur est que les affordances peuvent fonctionner selon différents niveaux de granularités d'interactions.

Cette granularité est tout aussi importante au niveau de la reconnaissance d'objets, puisqu'une détection d'affordance fine nécessite l'utilisation d'un classificateur capable de classifier différentes catégories et les différentes composants des objets appartenant à ces catégories afin de reconnaitre les interactions fines. Toutefois, ce type de classificateur nécessite un ensemble d'entrainement permettant de classifier les objets par catégories et par leurs sections, ce qui peut rendre encore exponentiellement plus longue la tâche de création d'un ensemble de données d'apprentissage pour un classificateur à apprentissage profond. Une approche simplifiée pour modéliser les affordances est de considérer l'interaction directe entre la personne et l'objet sous tout formes comme une activité seule.

3.4 Motivations et objectifs du mémoire

Le but principal de ce mémoire est d'adapter la reconnaissance d'affordances visuelles humain-objet afin de la rendre fonctionnelle pour un robot assistant mobile à ressources limitées. L'affordance a déjà été mainte fois analysée en relation au problème d'anticipation d'activités par un agent robotique, mais il existe peu de recherches sur la minimisation de la puissance de calcul nécessaire pour son intégration à un agent robotique. Un agent robotique doit conduire plusieurs systèmes critiques en parallèle, tels que le module de déplacement, le système de recharge, le module de reconnaissance de plan, etc. Il est donc intéressant de se pencher sur l'implémentation d'une solution capable de rouler des modules avec peu de ressources, par exemple en utilisant un ordinateur portable sans carte graphique, ce dernier composant étant cher et utilisé par plusieurs chercheurs pour exécuter le processus de classification en un temps raisonnable. Dans un tel cas, plusieurs approches telles que la modélisation du champ visuel en nuages de points 3D utilisent trop de ressources critiques pour d'autres systèmes clés et ne sont pas aptes à être implémentées.

Un autre constat est que, puisque ce projet est en association avec le groupe AGE-WELL, certains critères et besoins sont fixes afin de pouvoir rendre l'assistant robotique abordable pour la production et pour la vente. Afin de pouvoir atteindre ces besoins, le système de reconnaissance d'activité doit pouvoir communiquer aisément ces informations à un module de reconnaissance de plan haut-niveau en utilisant seulement un capteur RGB-D connecté à la plateforme robotique mobile.

Afin d'atteindre ces critères, l'objectif fixé est la création d'un système de reconnaissance d'activité qui peut être aisément intégré aux composants du robot. Plus particulièrement, le système de reconnaissance d'activité doit, par le biais de son capteur RGB-D, apprendre à reconnaitre et à profiler les objets et les personnes situées dans l'environnement en partant d'une base de connaissance générique. Celle-ci sera peaufinée au fil du temps selon les objets de l'utilisateur, afin de pouvoir déterminer les activités en cours d'exécution et leurs durées probables qui peuvent ensuite être envoyées à un module de reconnaissance de plan. Tout ceci sert à rendre l'agent robotique plus apte à détecter et profile les actions de son utilisateur. En d'autres mots, le système doit pouvoir offrir un module de reconnaissance d'activités efficient basé sur la reconnaissance d'affordances humain-objet. Ce module doit avoir la capacité d'auto-amélioration au fil du temps afin de pouvoir détecter de nouveaux objets précédemment inconnus aux systèmes et des objets existants découverts sous de nouveaux angles. Finalement, le module doit interagir avec un planificateur tout en étant capable de fonctionner avec les ressources d'un ordinateur portatif.

CHAPITRE IV

INTÉGRATION DE LA RECONNAISSANCE D'ACTIVITÉ PAR DÉTECTION ET CLASSIFICATION DE FORMES POUR UN ROBOT ASSISTANT

Dans ce chapitre, on présente notre approche pour effectuer la tâche de reconnaissance d'activité par vision artificielle pour un robot d'assistance cognitif. On y présente la conception et l'implémentation d'un module de vision et de reconnaissance d'activités. Ce module est destiné à être intégré à une plateforme robotique mobile afin de localiser la personne âgée que l'on souhaite assister et de reconnaitre les activités qu'elle effectue dans sa vie quotidienne.

Tel que mentionné précédemment, ce module est destiné à être connecté avec un module de reconnaissance de plan, conçu à l'externe par un autre étudiant associé au projet de **AGE-WELL**. Le module résultant de la combinaison des deux se nomme **PARC** (*Plan and Activity Recognition Component*). Le système PARC est conçu pour être intégré facilement à un robot afin d'effectuer, grâce à une phase d'extraction et de classification des caractéristiques visuelles d'une image, les activités en cours d'exécution pour ensuite inférer qu'elle est le plan primaire derrière cette action. Par exemple, le système doit être capable d'extraire d'une image en entrée l'activité en cours, par exemple *tenir une assiette*, pour ensuite transférer cette information aux autres modules du robot.

Le reste de ce chapitre est découpé selon les sections suivantes. On commençe avec les choix de conception et d'implémentation de PARC, tels que les capteurs d'observation nécessaires pour le fonctionnement du système. Ensuite, on discute des modules de détection et de reconnaissance des objets de l'environnement de PARC. La section est découpée en deux parties, la première décrivant la conception d'un module d'auto-apprentissage permettant à PARC d'augmenter ses bases de connaissances sur le foyer de l'utilisateur et les objets qui s'y trouvent, et ce, afin de raffiner ses détections d'activité à long terme. Cette approche se base sur un algorithme qu'on a nommé la **recherche sélective par profondeur**, conceptualisée afin de détecter des formes inconnues au système et de les ajouter à sa base de connaissances. On continue ensuite avec une description du module de détection des objets de l'environnement observé par la caméra de l'agent, ainsi que la position de l'utilisateur observé. Une sous-section décrit le modèle du réseau neuronal convolutif avec régions, implémentées selon les normes de la famille d'architecture YOLO, qu'on utilise comme module de classificateur d'objets afin d'offrir une détection rapide d'objets dans un environnement aux ressources limitées. Le chapitre se poursuit avec une décription du module de reconnaissance d'activité selon la détection d'affordances humain-objet dans un environnement avec un haut taux de rafraichissement des données à la seconde. Finalement, on introduit brièvement le module de reconnaissance de plan conceptualisé par le second étudiant sur le projet, Jean Massardi, et son interaction avec le système de reconnaissance d'activité de PARC.

4.1 PARC : Implémentation et design

Figure 4.1: Interface du système PARC.

Le choix des différents composants, qu'ils soient physiques ou logiciels, de PARC se base sur une philosophie selon laquelle la reconnaissance d'activité et la reconnaissance de plan doivent s'exécuter en temps réel et efficient en termes de ressources computationnelles. Un robot assistant doit être prêt en tout temps à reconnaitre les activités et les plans d'une personne afin de lui offrir une aide appropriée au bon moment. Pour atteindre cet objectif avec PARC, on doit conceptualiser son design, ses stratégies, son intégration et même les capteurs physiques utilisés par le module pour assurer leur bon fonctionnement selon la philosophie choisie.

Après l'étude approfondie du domaine et des discussions avec divers membres du projet sur les besoins des clients potentiels, ainsi que sur les choix de capteurs utilisés lors du prototypage de la plateforme robotique finale, on conçoit PARC pour fonctionner dans un environnement à ressources limitées, afin de faciliter l'implémentation du module dans le système d'un robot assistant cognitif. Par environnement à ressources limitées, on entend plus spécifiquement que le système peut être implémenté sur des systèmes tels qu'un ordinateur portable de gamme moyenne pour exécuter le module, doté d'une plateforme robotique mobile effectuant les tâches de positionnement dans l'espace et de capteurs tels qu'une caméra,

pour collecter des données sensorielles. Le système est construit pour effectuer la détection et la reconnaissance des activités d'une personne âgée dans des environnements intérieurs (comme une maison), basés uniquement sur les données sensorielles d'un capteur RGB-D. Ces données sont ensuite communiqués sous forme d'une grammaire formelle aux autres modules du robot, incluant le module de reconnaissance de plan qui permet de prédire les activités probables à la détection future. Ces données d'activités et de plan peuvent être ensuite transférées aux autres modules du robot, qui sont implémentés par d'autres chercheurs associés au projet **AGE-WELL**, afin de mettre à jour son planificateur ; de raffiner ses choix de déplacement pour optimiser le positionnement de capteurs vis-à-vis des emplacements probables pour les activités futures à détecter ; et de choisir les actions à entreprendre pour apporter une aide à son propriétaire.

Pour les fins de ce mémoire, seuls les modules de reconnaissance d'activités et de plans sont décrits.

4.2 Stratégie de localisation d'objets

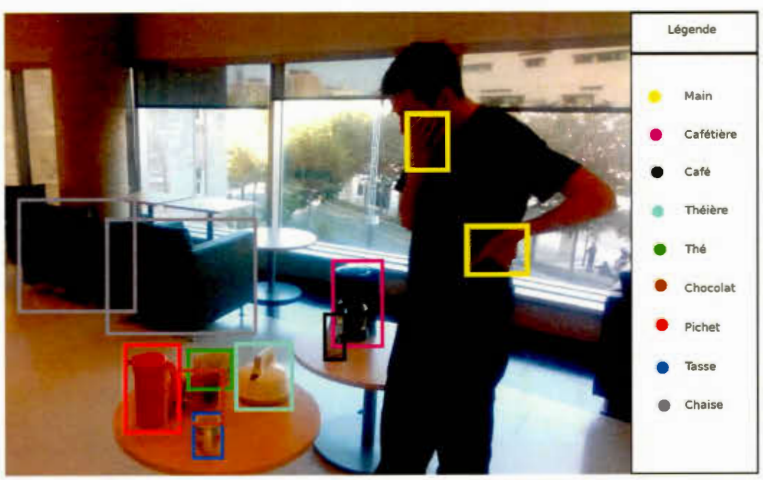


Figure 4.2: Exemple de zones de démarcation d'objets.

Le choix de la stratégie principale pour effectuer une localisation des objets candidats dans les données de traitement est passé par plusieurs itérations. Afin de pouvoir suivre les choix de conception définis précédemment, il est nécessaire que l'approche puisse offrir une localisation des régions candidates en temps réel en plus d'être capable de découvrir des régions d'objets non connus du système au moment actuel t_o. Ce dernier point est essentiel, puisque certaines catégories d'objets dans l'environnement peuvent ne pas être connu par le modèle de classification du robot actuel, mais sont toutefois pertinentes pour la reconnaissance d'activité. Par exemple, le système de reconnaissance plan peut détenir un plan **Préparer un café**, dont une des activités, **prendre conteneur de café**, peut être effectué par l'interaction d'une main avec diverses marques de café, qui ont toutes un format et un design unique. Puisqu'il est impossible d'entrainer un modèle de détection d'objet assez générique pour reconnaitre toutes les formes possibles dans le présent et l'avenir d'une catégorie d'objet, il est critique de permettre au module la possibilité d'apprendre de nouvelles informations de l'environnement afin de les ajouter à sa bases de connaissance.

On a originellement opté pour l'utilisation d'un réseau neuronal convolutif, couplé d'un algorithme de générations de régions pour déterminer les régions à classifier. Ce choix de conception se base sur l'idée que l'utilisation d'un module de génération de régions non supervisées dans une image pour la tache de détections de formes nous permet d'assurer au module la détection de catégories d'objets précédemment inconnus du système. Toutefois, cette approche s'est révélée inefficiente à cause du temps de traitement élevé des algorithmes de l'état de l'art pour la génération de régions sans connaissance préavis. Le découpage des régions d'une image 480×360 pixels avait une vitesse de traitement de 1 image par 5 secondes, ce qui est loin des besoins réels d'un robot assistant qui peut recevoir jusqu'à 30 images par seconde. On s'est ensuite tourné sur l'implémentation d'un réseau de neurones avec une sélection automatique de régions. Ceci nous permet

d'effectuer les phases de reconnaissance et de classification de formes en parallèle sur les données captées par notre caméra, le tout dans un temps de calcul raisonnable. Toutefois, cette approche ne nous permettait pas d'assurer au module de pouvoir reconnaitre l'addition de nouveaux objets, ou même de formes d'objets existants avec un haut taux de variance visuelle dans l'environnement. L'utilisation de nouveaux objets, telle qu'une nouvelle marque de lait lors de l'exécution du plan *Préparer un déjeuner*, peut causer une baisse du taux de reconnaissance jusqu'à un taux de 0%, puisque le modèle de classification n'a pas été entrainé pour le reconnaitre.

Le modèle est entrainé pour reconnaitre les différentes formes des classes d'objets dans l'environnement selon son ensemble de données d'entrainement, qui est limité par le choix de classes représentées et la qualité des données d'entrainement pour représenter ces classes sous différentes formes et selon différents critères de variances possibles. L'ajout de nouvelles données d'entrainement pour mettre à jour le modèle ne peut non plus s'exécuter en temps réel, puisque les modèles de classification par réseau de neurones par apprentissage profond nécessitent une nouvelle phase d'entrainement du modèle pour toute addition de classes supplémentaires. Cette limitation met l'emphase sur les deux points critiques essentiels du module : comment déterminer les régions d'objets potentiels dans une image sans nécessiter des connaissances à priori ? Et comment assurer en même temps que le système donne une réponse en temps réel lorsque les approches de ce domaine sont voraces en ressources et en temps d'exécution ?

Pour la première question, l'utilisation d'une approche de génération de régions basée sur les techniques d'image scoring en supplément au classificateur permettrait d'assurer à PARC de détecter des formes inconnues à son modèle, si on met de côté la problématique du temps de calcul élevé pour ces approches. Lors de la reconnaissance de régions, PARC nécessite de pouvoir définir le plus grand nombre

de régions possibles classifiables. Son système de reconnaissance d'activités, décrit avec plus de détails dans la section 4.4, fonctionne selon la reconnaissance fine d'affordances humain-objet. Dans ce cas, il est important d'assurer que chaque objet distinct est reconnu, puisqu'un objet manqué signifie par conséquent une activité manquée et ainsi un plan manqué. Il est toutefois peu intéressant de générer toutes les régions potentielles dans une image afin d'assurer l'optimalité des détections, puisque cette approche peut dégénérer à la génération de toutes les régions possibles, soit $(1..m) \times (1..n)$, où m et n sont les dimensions de l'image d'entrée, pour un jeu d'images reçu à un rythme de 30 Ips (images par seconde).

Après maintes itérations, le choix de conception pris pour effectuer cette génération de régions est l'implémentation d'un algorithme de générations de régions, adapté à nos besoins. L'algorithme de base qu'on a choisi est **Selective Search** (Uijlings *et al.*, 2013), une approche qui offre un bon ratio entre la qualité de la détection et la vitesse de calcul (Hosang *et al.*, 2014). L'algorithme a été toutefois modifié selon les besoins de notre système, afin d'offrir des détections de régions d'objets plus fines. À sa base, Selective Search est conçu pour fonctionner dans un environnement à deux dimensions, puisque la plupart des ensembles de données en imagerie numérique contiennent seulement des représentations en deux dimensions des objets Le module PARC, quant à lui, détient en tout temps des données de profondeur de son environnement grâce à son capteur de vision d'images de couleur et de profondeur. Les données de distances entre l'objet et son environnement sont riches en caractéristiques qui sont utiles pour représenter les différentes régions d'une image. On a alors conceptualisé une extension de l'algorithme afin de traiter ces caractéristiques lors d ela phase de détection.

L'algorithme résultant, nommé **Recherche Sélective par Profondeur**, est décrit en plus de détails dans la section 4.2.1. Son utilisation permet de mettre l'accent sur la reconnaissance d'objets à caractéristiques inconnues par le système.

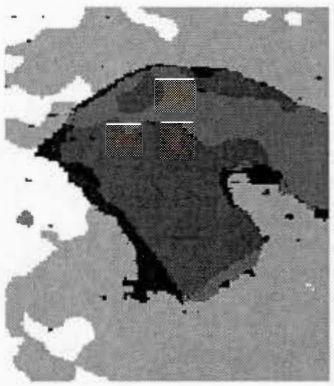


Figure 4.3: Image de profondeur d'une main tenant un pichet devant un fond blanc représenté par des nuances de gris. La différence de profondeur entre l'objet et l'arrière-plan est significativement distincte et permet de détecter ses contours, même sans l'utilisation des pixels de couleurs.

À long terme, ce type d'approche permet au système de détecter de nouvelles catégories d'objets introduits par l'utilisateur dans l'environnement, de détecter de nouvelles représentations en deux dimensions d'objets précédemment inconnus au module, ou même de reconnaitre de nouveaux types d'activités humain-objet précédemment inconnues dans la bibliothèque de plans de PARC. Au fil du temps, le robot assistant doit pouvoir reconnaitre des activités inconnues de sa bibliothèque de plans, que ce soit pour un processus de détection des oublis de l'utilisation lors de l'exécution des activités d'un plan, possiblement dû à une perte cognitive, ou pour permettre au module de découvrir l'addition de nouvelles activités précédemment inconnues, afin qu'un système expert puisse utiliser cette information pour mettre à jour la bibliothèque de plans. Toutefois, même si le temps de calcul de la recherche sélective par profondeur est rapide lorsqu'on le compare à d'autres approches du même domaine, elle est loin de pouvoir offrir un résultat plusieurs fois par seconde. Lors des tests préliminaires, l'algorithme prend environ $5.5 - 7$ secondes pour générer les régions d'une image de résolution **480p**.

Afin d'assurer que PARC puisse effectuer ces calculs dans des délais raisonnables on opte pour l'utilisation de deux systèmes de reconnaissance mis en parallèle, le premier étant la recherche sélective par profondeur. Pour assurer au module de pouvoir effectuer la reconnaissance ainsi que la classification des objets de l'environnement dans un temps raisonnable, PARC est doté du second système, un classificateur d'objets avec une sélection automatique de régions basée sur l'architecture YOLO, décrite en détail dans la section 4.3. Le classificateur utilise un ensemble de données d'entrainement, composé d'une liste d'images sous format JPEG associées à des fichiers TXT contenant le positionnement des différents objets classifiables à l'intérieur de l'image. L'emplacement de ces régions dans l'ensemble d'apprentissage est utilisé lors de son entrainement pour apprendre les différentes positions susceptibles de détenir un objet spécifique dans le champ visuel. L'utilisation d'un classificateur doté de la capacité de déterminer lui-même les régions de l'image à analyser nous permet ainsi de découper la phase de détection et de classification des objets en deux modules séparés : le **système d'autoapprentissage des connaissances** et le **système de classification d'objets**.

4.2.1 Autoapprentissage des connaissances du robot assistant

Le module d'autoapprentissage sert de système de tolérance à l'erreur. Pendant que le second module de classification effectue le processus de détection et de classification d'objets en temps réel, le système d'autoapprentissage effectue une passe sur une image captée avec **l'algorithme de recherche sélective par profondeur**. Ce module peut ensuite utiliser les nouvelles régions calculées pour effectuer une classification externe à PARC afin de confirmer si ces régions détiennent des objets inconnus au classificateur local, et en ce cas, si on peut alors les ajouter à l'ensemble de connaissances afin de mettre le classificateur à jour lors du prochain entrainement de son modèle de réseau de neurones.

L'algorithme, présenté dans le pseudo-code 1, est une extension de l'algorithme **Selective Search** pour permettre l'utilisation des caractéristiques de disparités de profondeurs lors du calcul de la similitude des différentes propositions de régions. (Uijlings *et al.*, 2013; Ikemura et Fujiyoshi, 2010). Cette donnée a un caractère plus invariant aux changements de couleurs et de textures que l'approche originale de la recherche sélective. En effet, la carte de disparité de profondeur d'une caméra RGB-D offre des résultats consistants, peu importe le niveau de luminosité appliqué sur une image, par exemple si la salle a des lumières allumées ou fermées.

Algorithme 1 : Recherche sélective par profondeur. Basé sur (Uijlings *et al.*, 2013).

Entrée : Matrice couleur (RGB) et matrice de profondeur.

Paramètre : échelle : Ratio d'échelle pour les caractéristiques de l'image.

Paramètre : coupeMin : Taille minimale d'une découpe

Paramètre : tMin : Taille minimale d'une région

Paramètre : tMax : Taille maximale d'une région

Paramètre : err : Ratio limite entre largeur et profondeur

Sortie : Vecteur de zones prometteuses R

1 **début**

2 Découpage de l'image en régions par une approche de segmentation de graphes. $R = seg(im_{rgb}, échelle, coupeMin)$

3 Vecteur de similitude $S = \emptyset$

4 **pour chaque** $paire(r_i, r_j)$ **faire**

5 Calcul similitude de couleur (r_i, r_j) sur im_{rgb}

6 Calcul similitude de texture (r_i, r_j) sur im_{rgb}

7 Calcul similitude de taille (r_i, r_j) sur im_{rgb}

8 Calcul similitude de distances (r_i, r_j) sur im_{rgb}

9 Calcul similitude de profondeur (r_i, r_j) sur im_{pf}

10 Calcul similitude de distance de profondeur (r_i, r_j) sur im_{pf}

11 Fusion des similitudes $s(r_i, r_j)$

12 $S = S \cup s(r_i, r_j)$

13 **tant que** $S \neq \emptyset$ **faire**
14 extraction de $max(S)$
15 Fusion $r_t = r_i \cup r_j$
16 $S \setminus ((r_i, r_\star) \cup (r_\star, r_j))$
17 $S_t = \emptyset$
18 **pour chaque** $paire(r_t, r_\star)$ **faire**
19 Calcul similitude de couleur (r_t, r_j)
20 Calcul similitude de texture (r_t, r_j)
21 *... Recalcul des valeurs de la ligne 4.*
22 $S = S \cup S_t$
23 **si** *ratio* $r_t(im_{rgb}).x$ / $r_t(im_{rgb}).y \leq err$ **alors**
24 **si** $taille(r_t(im_{rgb}) > tMin$ **et** $taille(r_t(im_{rgb}) < tMax$ **alors**
25 $R = R \cup r_t$
26 **retourner** R

L'algorithme prend en entrée Im_{rgb} et Im_{pf}, respectivement les matrices d'images de couleur et de profondeur de l'environnement et donne en sortie un vecteur de régions d'objets probables R. Le système démarre de la même manière que l'algorithme de recherche sélective, c'est-à-dire par le découpage de l'image en différents segments par une approche de segmentation de graphe (Felzenszwalb et Huttenlocher, 2004).

La matrice couleur de l'image est ensuite clonée afin de la transformer en différents espaces de couleurs avec des caractéristiques plus invariantes que l'espace RGB, tels que HSV et Lab. Ces matrices, ainsi que la matrice de profondeur sont envoyées aux fonctions de similitude de régions de couleur, ainsi que celles de profondeur.

$s_{profondeur}(r_i, r_j)$ calcule une mesure de similarité entre les variations de profondeurs de deux régions : r_i et r_j. Pour ce faire, la fonction transforme les valeurs de profondeurs de chaque région en état de couleur de type *nuances de gris*. Ces matrices sont ensuite utilisées pour calculer un histogramme d'une dimension sur le canal de gris (Ikemura et Fujiyoshi, 2010), une méthode conventionnelle pour représenter les valeurs de profondeur dans une image, avec une amplitude de 16 et une normalisation L_1. La similitude des deux régions est ensuite calculée par la formule :

$$s_{profondeur}(r_i, r_j) = \sum_{k=1}^{n} \sqrt{c_i^k, c_j^k} \tag{4.1}$$

$s_{profondeurDist}(r_i, r_j)$ calcule une mesure de similarité entre la profondeur moyenne des régions r_i et r_j. Le raisonnement derrière cette mesure est similaire à celle du calcul de similitude de distances, c'est-à-dire que deux régions sont plus propices à être fusionnées si elles ont une intersection prometteuse. Pour ce faire, la fonction calcule la profondeur moyenne des sous-matrices de profondeurs des régions r_i, r_j ainsi que leur union $r_{i \cup j}$. L'équation peut alors utiliser ces valeurs pour déterminer le niveau de différence de profondeur qui sépare les deux zones.

$$s_{profondeurDist}(r_i, r_j) = 1 - \frac{dist(r_{i \cup j}) - dist(r_i) - dist(r_j)}{dist_{max}} \tag{4.2}$$

La valeur est divisée par la profondeur maximale possible pour le capteur afin d'assurer une certaine consistance avec les équations de similitude de couleur.

Ces valeurs sont ensuite ajoutées à la formule initiale de la recherche sélective, où $a_i \in 0, 1$.

$$s(r_i, r_j) = a_1 s_{couleur}(r_i, r_j) + a_2 s_{texture}(r_i, r_j) + a_3 s_{taille}(r_i, r_j) + a_4 s_{distance}(r_i, r_j) + a_5 s_{profondeur}(r_i, r_j) + a_6 s_{profondeurDist}(r_i, r_j) \quad (4.3)$$

Après plusieurs phases d'optimisation pour trouver les valeurs de normalisation optimales pour les fonctions, les valeurs de chaque a_i ont été fixées de la manière suivante : $a_1 = a_2 = 0.8$,$a_3 = 0.75$,$a_4 = 0.9$ et $a_5 = a_6 = 0.95$.

Après avoir calculé les degrés de similitudes entre chaque région, l'algorithme démarre l'étape finale de la fusion des régions avec un haut degré de similarité. Pour ce faire, on extrait de S la plus grande valeur de similitude s_t ainsi que ses régions r_i, r_j afin de les fusionner et ensuite retirer les régions $s_k \in S$ qui utilise soit r_i ou r_j. Une fois le nettoyage terminé, on calcule le degré de similitude entre la nouvelle région et celles restantes et on les rajoute dans S. Finalement, une vérification sur r_t est effectuée pour confirmer qu'elle n'est pas de taille peu significative et qu'elle ne détient pas un ratio peu intéressant pour déceler un objet avant de l'ajouter au jeu de sortie. (Par exemple une région de ratio 30×2 peut être intéressante lors des fusions des régions intermédiaires, mais a peu de chances de détenir un objet classifiable.)

Après avoir calculé les hypothèses de région par la recherche sélective par profondeur, PARC effectue un recadrage sur l'image d'origine afin de créer n sous-images, où n est le nombre d'hypothèses de régions. Lors d'une période d'entraînement automatique, PARC transfert chacune des sous-images à un classificateur externe afin d'effectuer la tâche de classification.

Comme décrit précedemment, idée derrière cette couche de communication additionnelle est qu'il est impossible pour un robot cognitif d'avoir un jeu de données en même temps spécifique à l'environnement de son propriétaire tout en étant

assez complexe pour reconnaitre toutes les possibilités de formes d'objets dû aux différents facteurs de variance. Créer un tel type de modèle pour une approche à apprentissage profond nécessiterait un jeu de données massif, dans l'ordre de millions de photos pour chaque objet qui pourrait être retrouvé dans l'environnement sous chaque angle possible, pour chaque degré de luminance possible, etc. Ce type de ressources est tout simplement hors de la portée de PARC. Toutefois, puisque le module d'auto reconnaissance n'est pas limité par le temps de traitement du reste du système, il lui est possible de communiquer avec une seconde base de connaissances qui est entrainée sur un plus grand ensemble de données, mais dont le temps de réponse est significativement plus lent.

PARC envoie par conséquent ses régions potentielles au second classificateur externe et reçoit le nom de l'objet situé dans cette région ainsi que son taux de confiance.

Ces données peuvent ensuite être transférées, avec une copie du résultat du classificateur local sur l'image d'origine, à l'algorithme 2 pour déterminer quelles régions sont les plus prometteuses à ajouter à l'ensemble de connaissance d'entrainement du classificateur de PARC.

Algorithme 2 : Vérification de redondances de régions.

Entrée : R : Vecteur de classes avec régions du système expert et R_l : vecteur de classes avec du classificateur local.

Paramètre : conf : Niveau de confiance minimal

Sortie : R_{nouv} : Vecteur de régions uniques à R.

```
1  début
2      R_nouv = ∅
3      ratioUnique = 1
4      pour chaque Région r dans R faire
5          pour chaque Région r_l dans R_l faire
6              red = regUniques(r, r_l)
7              ratioUnique = min(ratioUnique, red)
8          si ratioUnique > conf alors
9              R_nouv ∪ r
10             R \ r
11         ratioUnique = 1
12     retourner R_nouv
```

$$regUniques(reg_{glo}, reg_{loc}) = \frac{4}{9}(1 - s_{profondeurDist}(reg_{glo}.region, reg_{loc}.region)) + \frac{2}{9}(1 - s_{distance}(reg_{glo}.region, reg_{loc}.region)) + \frac{1}{9}levNorm(reg_{glo}.id, reg_{loc}.id) + \frac{2}{9}reg_{glo}.confiance \quad (4.4)$$

L'équation 4.4 définit le taux de rapprochement de deux propositions de classes pour une région. Les poids $\frac{4}{9}$, $\frac{2}{9}$, $\frac{1}{9}$, $\frac{2}{9}$, ont été fixés empiriquement selon des tests en cours de développement afin d'assurer l'importance des similarités de profondeur vis-à-vis les valeurs de similarité de celles de la recherche sélective traditionnelle.

La fonction est paramétrée sur $regUniques(reg_{glo}, reg_{loc}) \in [0, 1]$ et reg_{glo}, reg_{loc}, qui détient pour chacun la région de l'objet, son nom de classe, son identifiant numérique et le pourcentage de confiance que la région détient réellement cedit objet. Les deux premières vérifications, $s_{profondeurDist}$ et $s_{distance}$, sont similaires à celles de la recherche sélective par profondeur, mais ont une inversion de valeur puisque la fonction calcule le ratio de différence entre les régions plutôt que leur ressemblance.

$levNorm$ est une version modifiée de la fonction de distance de Levenshtein avec une normalisation basée sur la taille maximale des noms de classe en entrée. (Il existe des processus de normalisation pour la distance d'édition qui suivent de plus près les principes de la **mesure de similarité**, telles que (Yujian et Bo, 2007), mais elles ne sont pas utiles dans ce contexte, puisqu'on utilise la distance pour traiter le cas d'erreur où les deux classes ont le même nom, lorsque l'une d'entre elles détient une erreur typographique.) La dernière valeur prise est le niveau de confiance de l'objet découvert.

$$levNorm_(clA, clB) = \frac{lev_{clA,clB}(x, y)}{max(longueur(clA), longueur(clB))} \tag{4.5}$$

L'algorithme 2 calcule le taux d'unicité de chaque région détecté par l'algorithme 1 par rapport aux régions du classificateur local. Il garde ensuite le plus petit taux calculé, puisque nous voulons confirmer si la région est unique, même lorsque comparée à celles qui lui sont très similaires. Si le niveau d'unicité minimale est supérieur au taux de confiance $conf$, alors la région, ainsi que sa classe, est ajoutée au jeu d'ensemble de l'image d'originale. Celui-ci est ensuite transféré aux images d'entrainement du réseau local.

Le module est pensé pour pouvoir démarrer un nouvel entrainement de son modèle de classification lorsque le robot est en période de pause, par exemple, lorsque le

robot est en période de recharge pour sa batterie. Lors de l'étape du transfert, PARC peut regarder si l'une des nouvelles régions détient un objet de classe actuellement inconnu du système et met le modèle à jour en conséquence.

Si une nouvelle classe a été découverte, alors le robot démarrera alors un nouvel entrainement de son modèle en partant de zéro. Par contre, si chaque région est de classe connue et que les nouvelles données d'entrainement sont seulement des caractéristiques d'objets connus, alors PARC garde le modèle précédent comme base et le système raffine uniquement ses poids en redémarrant son entrainement.

4.3 Stratégie de classification

Comme décrit dans les sections précédentes, PARC détient un second système de classification d'objets qui s'exécute en parallèle avec le module d'autoapprentissage. On utilise ce module pour effectuer la reconnaissance des objets dans l'environnement, ainsi que des mains d'un utilisateur.

Ce choix de conception nous permet d'éviter d'avoir recours à un module additionnel pour la reconnaissance du corps humain, puisque notre approche de détection d'activités est basée uniquement sur les affordances mains-objets.

Ce module est implémenté par un réseau neuronal convolutif avec découpage de régions. Comparativement aux approches de générations de régions individuelles pour la détection de formes, le réseau de neurones par apprentissage profond permet de fusionner les tâches de détection et de classification dans la même architecture. Ceci permet d'éviter une redondance de calculs lors du processus de classification pour les cas où les zones d'ancrage ont un haut degré de similarité.

Le module est passé à travers maintes itérations. La première implémentation a eu comme base le modèle *AlexNet* implémentée par la bibliothèque Caffe avec une utilisation de l'algorithme de recherche sélective par profondeur pour la génération

de régions. Cette approche avait toutefois de la difficulté à détecter les interactions humain-objet dues aux limitations du modèle d'Aletnex pour des caractéristiques de bas niveau. Un autre problème de cette approche est que l'approche de génération de régions nécessitait un temps de traitement d'environs $\approx 7-9$ secondes, ce qui est inacceptable pour les besoins du mémoire.

Le modèle du réseau a pris plusieurs autres formes lors de développement du mémoire, tel qu'un modèle basé sur les architectures R-CNN, Faster-R-CNN, Mask R-CNN, avant d'être fixée à un modèle hybride, décrite dans la section 4.3.1.

Le système reçoit en entrée une image couleur im. Cette entrée est redimensionnée en longueur et largeur de taille 284×284. Ces valeurs ont été fixées empiriquement lors de la phase de tests du système, puisqu'elle permet d'offrir un haut taux de précision dans un temps de calcul d'environ 16 Ips. La taille des images peut toutefois être agrandie jusqu'à une résolution de 384×384, la taille maximale d'images utilisées lors de l'entrainement du réseau. L'image redimensionnée est ensuite transférée au réseau neuronal qui retourne une matrice de cellules. On utilise alors cette matrice pour détecter les cellules avec un taux de confiance supérieure à $conf$, une valeur fixée empiriquement à 60 %.
Pour chaque cellule avec un taux de détection supérieure à $conf$, on encapsule l'information reçue dans une classe *DetectedObjet*. Cette structure contient le nom de la classe d'objets, son taux de confiance, la matrice de distance de la région et la zone d'ancrage de l'objet dans l'image d'origine selon les paramètres suivants : $pointCentral_x, pointCentral_y, largeur, longueur$. PARC transfère finalement le vecteur de régions détectées au module de reconnaissance d'activité.

4.3.1 Modèle du réseau de neurones

Comme décrit dans la section précédente, le modèle du réseau est passée à travers plusieurs phases afin d'offrir un haut taux de précision tout en utilisant le moins

de ressources possibles. Pour ce faire, notre implémentation est conçue pour s'exécuter dans un environnement avec seulement un processeur et aucun GPU grâce à un choix de bibliothèque de réseaux de neurones à apprentissage profond optimisé pour fonctionner sur un processeur, lequel est décrit plus en détail dans le chapitre suivant. Une carte graphique est tout de même utilisée lors de l'entrainement du réseau, mais n'est pas nécessaire lors de la tâche de classification.

Plusieurs modèles telles que GoogleNet, VGG ou Darknet ont été essayées, mais ces modèles n'offrent pas de bons résultats lorsqu'elles sont utilisées sans carte graphique. Toutefois, les modèles conventionnelles plus concises, telles que AlexNet ou cpuNet [1] n'atteignent pas un taux de détection supérieur à 50 % sur nos jeux de données lors de la phase des tests préliminaires.

Le modèle final conceptualisée pour le réseau de PARC est largement inspirée des modèles YOLOV3 et YOLOv3-tiny (Redmon et Farhadi, 2018). Le modèle YOLOv3, aussi connue sous le nom de Darknet, n'est pas à sa base un modèle efficiente pour fonctionner dans un environnement sans carte graphique, avec une vitesse de traitement d'environ 2-3 Ips pour notre ensemble de tests, lorsque nous l'avons essayé avec sa bibliothèque officielle.

Même si le modèle semblait peu optimal pour nos besoins, sa forme simplifiée, DarknetV3Tiny, quant à elle, offre une vitesse de classification supérieure à 20Ips sur notre ensemble de données avec un taux de classification plus élevé que celles proposées en ligne comme étant optimisées pour le CPU. Elle atteint selon nos tests initiaux 58 %. Toutefois, ce taux d'erreur de classification est encore trop élevé pour être utilisable selon les besoins de PARC. On propose alors une approche hybride afin d'optimiser la qualité des caractéristiques reconnues par le réseau.

---

1. `https://github.com/Guanghan/cpuNet`

La structure finale est basée principalement sur l'utilisation de couches de convolution, de suréchantillonage et de raccourcis. L'utilisation des couches raccourcies permet à notre réseau de concaténer les gradients des couches de niveau supérieur à celles de niveau inférieur, ce qui permet la création d'un modèle plus riche sans nécessiter l'ajout de couches et de paramètres additionnels qui augmenterait davantage le temps de calcul.

Le modèle final, représenté dans la figure 4.4 est constituée de 58 couches de convolutions, d'une couche de ré-échantillonnage et de 22 blocs de résidus. La première couche, représentée dans la figure par la section A, est une couche de convolution qui prend en entrée les trois canaux de couleurs, rouge-vert-bleu, de notre image 284×284 et effectue un couche de 32 filtres de convolution, afin de découper les caractéristiques de l'image pour en 32 canaux filtrés. Le choix de redimensionner l'image selon la taille 284×284 a été fait pour maximiser le temps de calcul sur notre processeur de test. La taille d'entrée a été fixée empiriquement après plusieurs périodes d'optimisation pour permettre l'utilisation d'un bon ratio entre le taux de classification des images par notre réseau et la vitesse de calcul sur un processeur, tout en suivant les contraintes de YOLO qui fonctionne seulement avec des images d'entrées à dimension divisible par 32.

Une seconde couche de convolution est ensuite appliquée avec des hyperparamètres de profondeur 64 et un pas de 2 afin d'effectuer une réduction de la taille des images intermédiaires analysées, sans devoir nécessiter l'utilisation de couches de pooling, ce qui augmenterait la vitesse d'entrainement du classificateur sans diminuer son taux de classification. La coupe génère un jeu d'images intermédiaires de dimension 144×144 de 64 filtres. Afin de paramétrer notre réseau pour détecter plusieurs formes uniques de caractéristiques sur cette taille, on utilise ensuite deux couches de convolution de taille 1×1 et 3×3, afin d'utiliser des couches avec des tailles de filtres variants entre 32 et 64. On additionne ensuite leurs résultats de

leurs fonctions d'action à celles du niveau supérieur représentées dans l'image par la section B, afin d'améliorer leurs valeurs et de permettre au gradient de se propager au niveau supérieur du réseau. Cette approche est souvent réutilisée dans le réseau, afin de permettre de mettre à jour les couches de convolutions et de réduction de taille des caractéristiques plus rapidement et aussi de permettre d'améliorer le résultat par la réutilisation des valeurs des couches précédentes, qui serait normalement perdue au cours de leur propagation dans un réseau d'une telle taille. Finalement, nous appliquons un filtre de taille $256 \times 3 \times 3$ avec un pas de 2 pour diminuer de moitié la taille des images intermédiaires à analyser.

On utilise ensuite la même technique sur nos images intermédiaires, maintenant de taille 72×72, avant de les réduire de nouveau par l'utilisation d'une couche de convolution avec 256 filtres et un pas de 2.

Nos images intermédiaires, maintenant de taille 36×36 sont transférées à 8 séries successives de couches avec filtres de taille 1×1 et 3×3. Ces couches permettent d'entrainer notre réseau à détecter des caractéristiques de taille moyenne dans une image selon un grand nombre de caractéristiques. Cette approche, inspirée directement de son utilisation dans le modèle YOLOV3, nous permet d'atteindre un niveau de classification sur des objets qui étaient entièrement ignorés par les autres modèles précédemment implémentées, tel que la détection d'une poche de thé, un objet dont la petite taille était rendue presque impossible à classifier lors de l'utilisation des autres classifications dans nos itérations précédentes. Cette section se termine finalement par une couche de convolution avec 512 filtres avec un noyau de taille 3×3 et un pas de 2.

Une dernière étape d'apprentissage finale pour la détection de caractéristiques est effectuée, cette fois sur 8 séries de couches de convolutions sur des images intermédiaires de taille 18×18. Leurs résultats, ainsi que ceux du niveau supérieur

représentés dans la figure par la section E, sont finalement fusionnés avec un bloc de résidu pour ensuite être transférés aux blocs de convolution utilisés par YOLO afin d'effectuer le processus de classification finale.

Une autre particularité reprise de YOLOv3 est l'utilisation d'une couche additionnelle de résidus suivant la couche de classification afin d'effectuer une seconde étape de classification sur des caractéristiques de différentes tailles. Pour ce faire, le réseau propage les résultats de la première couche de convolution de la section H et effectue un ré-échantillonnage de taille $2\times$ afin de les retransformer en images intermédiaires de taille 36×36. Le réseau fusionne ensuite ces données avec celles de la section E pour permettre au réseau d'effectuer une classification sur les caractéristiques de dimension 36×36. Ces données sont ensuite passées au processus de préparation pour la classification de YOLO afin finalement d'effectuer cette classification.
Comparée au modèle original de YOLOv3 pour YOLO, notre modèle utilise moins de couches de convolution et n'effectue pas une autre étape de réduction des images intermédiaires afin d'entrainer une troisième couche de sortie. Les gains offerts par une classification à si petit niveau, qui dans notre cas serait d'après une dimension 9×9 due à une étape de réduction additionnelle, ne peuvent offrir une amélioration du taux de classification qui est déjà rendu possible par les niveaux supérieurs. Leur utilisation aurait pour seule conséquence l'ajout de calculs inutiles au temps de traitement des requêtes d'entrainement et de classification.

4.4 Stratégie de reconnaissance d'activité

On emploie un système de reconnaissance d'activité de bas niveau structuré selon une représentation des activités comme étant des affordances humain-objet.

Dans la version actuelle, le module de reconnaissance de plan et celui d'activité sont intrinsèquement reliés, le module de reconnaissance d'activités sert de pa-

ramètres d'entrée pour la reconnaissance de plan, lequel alimente le module de reconnaissance d'activités en retour par la prédiction des actions futures probable du plan.

La tâche de reconnaissance d'activité est effectuée par l'utilisation de deux structures préparées sur mesure, *GestionActivite* et *Affordances*. *GestionActivite* sert de cerveau pour l'entièreté de la tâche de reconnaissance pendant que *Affordances* est une base de connaissances sur les affordances perçues par le système.
La classe *Affordances* détient des instances d'une structure qui modélise les différentes instances temporelles des affordances, *AffordanceTemps*. *AffordanceTemps* contient lui-même une instance d'une structure *Activite*, qui détient le nom de l'activité (la modélisation des affordances directe humain-objet permet l'utilisation du nom de l'objet comme nom de l'activité.), la région où l'objet a été détecté par le capteur, le taux de confiance de cette détection et la distance moyenne de profondeur entre l'objet et le capteur.
Les autres éléments de *AffordanceTemps* sont le temps de départ de l'affordance détectée et son temps de fin. Ces données temporelles permettent d'assurer un suivi temporel pour l'affordance Aff_t basée sur les détections précédentes $Aff_k \in t_0..,t_{k-1}$.

Lors d'une phase de détection et de reconnaissances d'activités, PARC transmet les données sensorielles de son capteur au classificateur et reçoit un vecteur d'instances de la classe *DetectedObject*. Ce vecteur est transmis au module *GestionActivite*, qui les découpe en catégories **membres du corps humain** et **objets de l'environnement**. Dans PARC, on effectue la reconnaissance du corps humain en parallèle à la reconnaissance des objets de l'environnement par le module de reconnaissance d'objets afin d'éviter une redondance des calculs par l'utilisation d'algorithmes de détection du corps humain. Cette approche est fondée sur le principe que la plupart des activités de la vie quotidienne, hors certains cas spéci-

fiques tels que **dormir** ou **se laver**, nécessitent une interaction entre les mains de la personne et un objet de son environnement. Cette proposition est aussi pensée pour assurer une meilleure reconnaissance des affordances humain-objet, puisque le classificateur est entrainé sur un jeu de données de mains spécifiques à son propriétaire.

GestionActivite effectue une vérification d'affordances sur les différentes instances de *DetectedObject* pour les mains et les instances de *DetectedObject* spécifique aux objets. Cette fonction reçoit le positionnement des mains et des objets et met la base de connaissances des affordances en cours à jour. L'algorithme utilise alors l'équation 4.6 pour vérifier si une affordance est en cours.

$$\begin{aligned} affDetect(m,o) = (m.reg/capo.reg).aire() > \\ x_d min(m.reg.aire(), o.reg.aire())\&\& \\ |pf_{moy}(m.reg) - pf_{moy}(o.reg)| < y_d cm \end{aligned} \quad (4.6)$$

Pour confirmer une affordance, on vérifie que l'aire d'intersection des régions de la main et de l'objet, m et o, est supérieure à la plus petite des deux aires, multiplié par x_d, où x_d est un pourcentage minimal égal à 15 %. Cette valeur est fixé selon le niveau de contact habituel d'une main et des divers objets utilisés lors des phases de tests préliminaires du module. pf_{moy} dénote la profondeur moyenne d'une région. On confirme ainsi que la différence de profondeur entre o et m est plus petite que le seuil maximal $y_d cm$, qui a été fixé empiriquement à 10cm.

Si on détecte une affordance, la fonction vérifie si cette affordance représente le début d'une nouvelle activité, ou si elle est la continuation d'une activité déjà entamée. Cette confirmation nous permet de modéliser la temporalité d'une activité, ainsi que de traiter les cas de multiples reconnaissances d'affordances, puisque PARC est mis à jour plusieurs fois par seconde et qu'une interaction peut ainsi

être détectée plusieurs fois dans la même seconde. On vérifie ensuite si l'affordance est confirmée, c'est-à-dire si son taux de détections individuelles a dépassé le seuil nécessaire pour s'assurer que l'affordance perçue est réelle et n'est pas seulement un cas d'erreur. Ceci permet de traiter par exemple les cas où un utilisateur déplace certains objets pour accéder à son besoin primaire. Pour chaque affordance confirmée, on note officiellement sa confirmation et on l'ajoute aux affordances à communiquer avec les autres modules de PARC.

Algorithme 3 : Détection d'affordances en cours

Entrée : O : Régions de classe d'objets et M : Régions avec mains.

Sortie : Aff_t : Vecteur d' **AffordancesTemps** détecté.

```
1  début
2      pour chaque DetectedObject m dans M faire
3          pour chaque DetectedObject o dans O faire
4              si affDetect(m, o) alors
5                  si Affordances[o.classe].tempsfin() + delai < temps alors
6                      Affordances[o.classe].affFini()
7                  si Affordances[o.classe].affEnCours() alors
8                      Affordances[o.classe].demAff(obj, temps)
9                  sinon
10                     Affordances[o.classe].majAff(obj, temps)
11                 si Affordances[o.classe].confirme() alors
12                     Aff_t ∪ Affordances[o.classe]
13                     Affordances[o.classe].affActEnvoi()
14         ratioUnique = 1
15     retourner Aff_t
```

GestionActivite reçoit les nouvelles affordances détectées et effectue ensuite une phase de nettoyage. Lors de l'exécution d'une tâche de la vie quotidienne, il est possible de détecter deux activités en parallèle, telle que verser de l'eau d'une théière située dans la main gauche à une tasse située dans la main droite. Pour traiter ce cas, on vérifie que Aff_t est différente de $[Aff_{t-1}, Aff_{t-2}, Aff_{t-3}]$. La taille de 2 affordances passées a été fixée empiriquement afin de traiter le cas de deux activités parallèles dans un environnement avec occlusions possibles dû à un taux élevé de rapprochement des objets. *GestionActivite* confirme finalement que les affordances détectées sont dans la portée des actions prédites par le module de reconnaissance de plan avant de les lui transmettre. Dans une version future, cette information pourrait permettre de modéliser les comportements cognitifs étranges de l'inviddu.

4.5 Reconnaissance d'activité et reconnaissance de plan

Le module de reconnaissance d'activités est conçu pour communiquer directement avec un module de reconnaissance de plan, proposé par le second membre du projet VIGIL à l'Université du Québec à Montréal, le doctorant *Jean Massardi.* Ce module est un sujet de recherche distinct de ce mémoire et par conséquent n'est que décrit brièvement.

Le module de reconnaissance de plan de PARC utilise une bibliothèque de plan pour modéliser les plans de haut niveau, tel que défini par (Geib et Goldman, 2009).

Théorème 1 : Une bibliothèque de plans LP est un tuple $LP = (A, NT, G, P)$ où :

- A est un ensemble fini de symboles terminaux ;
- NT est un ensemble fini de symboles non terminaux ;
- $G \subset NT$ est un ensemble de buts
- P est un ensemble de règles de production sous la forme $\alpha \leftarrow S, \sigma$, où $\alpha \in NT$, S est un ensemble de $A \cup NT$ et σ un ordonnancement partiel de S.

Ce théorème peut être modélisé comme un arbre ET/OU hiérarchique partiellement ordonné. Cette forme permet ainsi de modéliser des buts de même niveau, une hiérarchie de buts et d'activités et permet de traiter le bouclage d'activités. Le module de reconnaissance d'activités transfère au module de la reconnaissance de plan les observations d'activités captées, qui utilisent une méthode par filtre à particule sur une bibliothèque de plans implémentés par une Grammaire non contextuelle partiellement ordonnée à multi ensembles.

Le module utilise ces observations pour mettre son modèle à jour et transmet au module de reconnaissance d'activité le plan probable actuel, ainsi qu'un ensemble des prochaines activités probables à détecter.

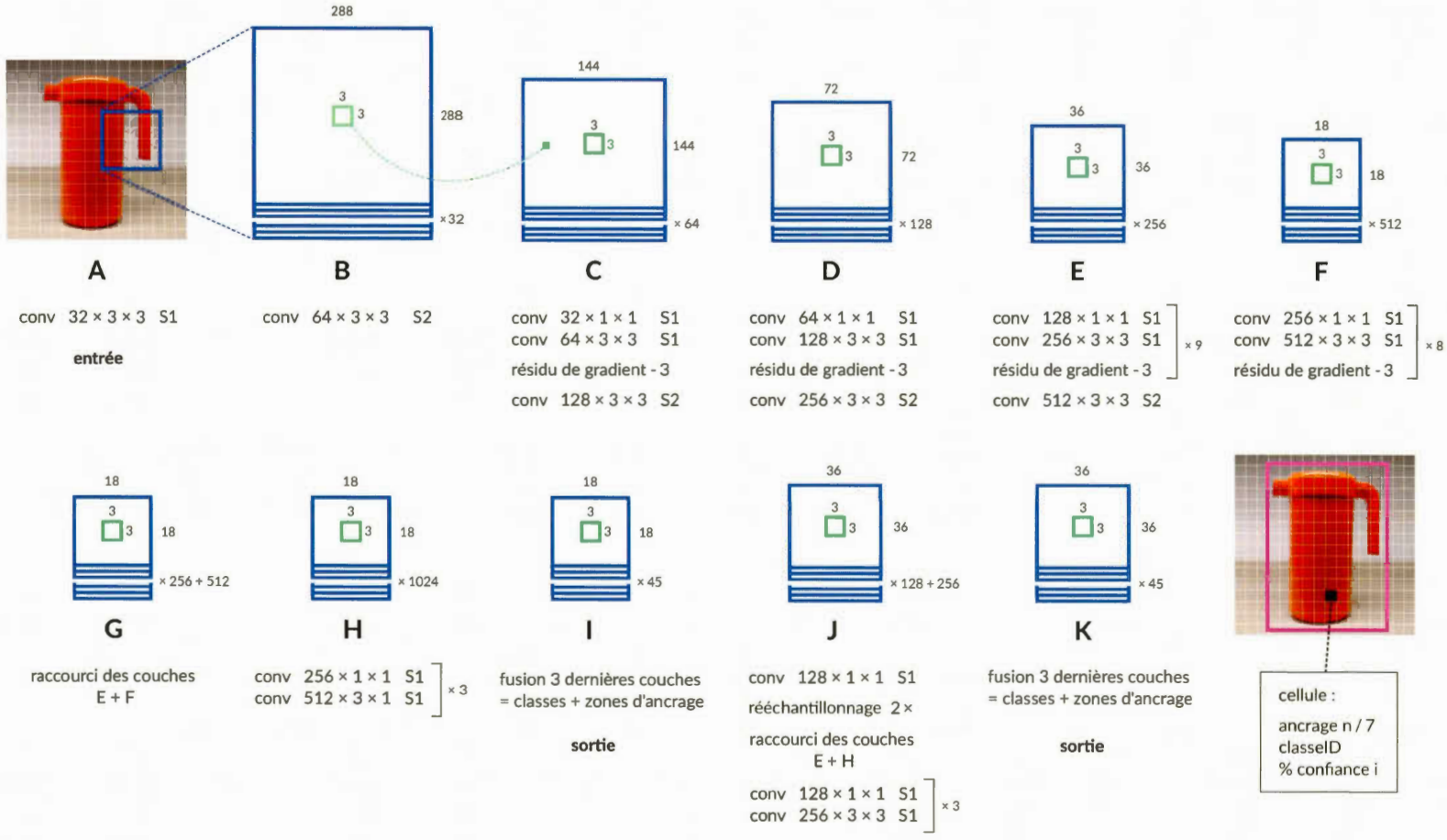


Figure 4.4: Modèle du CNN de PARC.

CHAPITRE V

EXPÉRIMENTATIONS ET ANALYSE DES RÉSULTATS

L'évaluation de PARC consiste en une phase d'expérimentation sur un corpus de reconnaissance de plan et d'activités. Cette expérience a pour but de confirmer la viabilité du système de reconnaissance d'activités en tandem avec son système de reconnaissance de plan pour reconnaitre l'exécution d'activités et de plans dans la vie quotidienne.

La phase d'expérimentation, décrite à la section 5.2, vise à reconnaitre des activités et leurs plans dans trois différents environnements contrôlés, ainsi qu'à détecter et confirmer l'avantage d'enrichir le modèle de classification au fil du temps par l'addition de nouveaux objets. Les résultats sont ensuite analysés selon diverses formes d'entrainement. Finalement, une analyse globale des résultats est effectuée avant de terminer sur les pistes d'évolution possibles du système lors de son intégration finale à un robot assistant cognitif.

5.1 Environnement d'évaluation

On a paramétré PARC afin de fonctionner sur un environnement d'agent robotique de gamme moyenne. Pour ce faire, nous avons implémenté le système afin de pouvoir fonctionner sur une plateforme robotique mobile *TurtleBot 2* par le biais d'un ordinateur portable *Asus Zenbook* doté d'un processeur *Intel Core i5 6200U*.

Figure 5.1: Système PARC déployé sur un robot TurtleBot 2.

Pour le choix de capteurs, la plateforme robotique a été dotée d'une caméra RGB-D *Intel RealSense D-435*.

Les modules de reconnaissance d'activités ainsi que ceux de reconnaissance de plan sont implémentés en C++. Ce choix de conception permet de faciliter leur intégration au système d'exploitation pour robots, *ROS*, un intergiciel offrant des logiciels et des bibliothèques facilitant la programmation de systèmes pour des plateformes robotiques.[1]. Le réseau neuronal convolutif YOLO a été entrainé sur un ordinateur *Intel Core i7 7700* avec une carte graphique *NVIDIA Geforce 1060*. Cet ordinateur est aussi utilisé comme banc d'essai pour PARC, puisqu'il nous permet de tester l'efficience de l'approche dans un environnement avec processeur

1. http://www.ros.org/

ainsi que dans un environnement avec une carte graphique lors de la phase de classification d'objets. L'implémentation du modèle du réseau de neurones pour s'exécuter sur un processeur s'est faite par l'intermédiaire du module DNN de la bibliothèque *OpenCV*[2]. Leur module d'implémentation de réseau à apprentissage profond est optimisé pour fonctionner sur CPU seulement et nous permet de garder la précision de notre réseau sans nécessiter l'utilisation d'une carte graphique.

À des fins de tests de l'approche de reconnaissance d'activités et de plan, nous avons décidé de créer notre propre corpus, dénommé le corpus *Drinkmaking*. Le choix d'utiliser un jeu de données personnel plutôt que ceux de l'état de l'art pour la reconnaissance d'activités est dû au fait que la plupart des corpus du milieu pour la reconnaissance de plan de haut niveau par reconnaissance d'activité de bas niveau ne sont pas adaptés à la problématique de PARC. Plusieurs corpus de détection d'activités, tels que *UCF 101*, ne représentent pas seulement des activités de vie quotidienne, ce qui engendre des contraintes d'environnement et de tests différents que ceux de PARC.

D'autres jeux de données pour la reconnaissance d'activités par affordance, tels que CAD-120, sont conçus avec leur propre grammaire formelle et nécessiteraient alors la réécriture de leurs plans dans leur entièreté. Le module de reconnaissance de plan de PARC fonctionne avec une grammaire syntaxique conçue sur mesure pour faciliter l'addition de nouveaux plans et d'activités par systèmes experts. En tant que première itération, elle n'est pas actuellement rétrocompatible avec la plupart des corpus de tests actuels, en raison de sa syntaxe et de sa démarcation des activités de bas niveau et de ses buts plus détendus. La syntaxe d'activité pour le corpus CAD-120, par exemple, décrit la séquence d'une activité pour chaque image individuelle des vidéos d'entrainement et a un ordonnancement linéaire fixe pour

2. `https://opencv.org/`

chaque activité. La gramme de PARC, quant à elle, décrit les suites d'activités dans un plan peu importe le temps et permet d'avoir un ordonnancement partiel sur l'ordre d'apparition des activités d'un plan.

Un corpus sur mesure a donc été créé afin de pouvoir mettre l'approche à l'épreuve sur un format de plans et d'activités adapté à sa syntaxe.

5.2 Évaluation de PARC sur Drinkmaking

Figure 5.2: Différents plans du corpus Drinkmaking

Le corpus PARC est conçu pour représenter l'exécution de différents plans de haut niveau par des suites d'activités dans un contexte d'actions de la vie quotidienne. Ce nouveau corpus est composé de diverses activités dans un environnement similaire à des salles que l'on peut retrouver dans n'importe quelle maison, n'importe quel appartement, etc. En comparaison de plusieurs corpus de reconnaissance d'activités et/ou de plan qui préconisent la classification d'actions élémentaires du corps humain, tel que **se lever**, **sauter** ou **marcher**, ou ceux qui utilisent des données sensorielles non visuelles, notre jeu de données est orienté sur l'utilisation de plans composés d'activités rapprochées dont certaines peuvent avoir différents ordres d'apparition pour un même plan.

Notre corpus, nommé Drinkmaking, est composé des plans, des actions et des classes suivantes :

{Tea-making.Hot-chocolate-making.Coffee-making}.

{Tenir}.

{Théière.Main.Thé.Tasse.Carton de lait.
Café.Cafetière.Chocolat.Pichet.Canette}

Voici quelques exemples de plans reconnaissables extraits du corpus :

- Chocomaking(<Hold(teakettle).
  Hold(Waterpitcher)>.Hold(Mug). Hold(Choco))
  Chocomaking(Hold(Mug).Hold(Milk). Hold(Choco))
- CoffeeMaking(Hold(coffee).Hold(coffeemaker).
  Hold(water).Hold(coffeemaker).Hold(mug).
  Hold(milk).Hold(mug))

Chaque plan représente un ensemble d'activités de bas niveau relié au plan. Chaque activité doit être exécutée en ordre, exceptées celles placées à l'intérieur des marqueurs <>. Cette démarcation permet d'avertir le modèle que chacune des activités contenues à l'intérieur peut apparaitre lors de l'exécution du plan dans n'importe quel ordre. Cette démarcation nous permet alors de traiter, dans le cas de <Hold(teakettle).Hold(Waterpitcher)>. Hold(Mug) et Hold(Choco), les séquences d'activités :

- Hold(teakettle).
  Hold(Waterpitcher).Hold(Mug). Hold(Choco)
- Hold(Waterpitcher).
  Hold(teakettle).Hold(Mug). Hold(Choco)

Pour bâtir le corpus, 10 vidéos d'évaluation et d'entrainement ont été enregistrées. Ce jeu de données représente l'exécution de diverses activités de la vie quotidienne telles que décrites dans le corpus Drinkmaking. Les plans ont été exécutés dans différentes salles d'un bâtiment avec des variations sur le positionnement et les angles de vue du capteur de données de PARC. Ce choix a été fait afin d'assurer au module de reconnaissance d'objets un entrainement basé sur les objets et les actions spécifiques au corpus, sous plusieurs formes propres à chaque vidéo afin d'extraire des caractéristiques plus riches pour chaque classe d'objets.

Chaque vidéo du corpus a été enregistrée avec une caméra RGB-D *Realsense 2 D-435* qui a généré deux vidéos, une de couleur et une de profondeur, toutes deux d'une résolution de 1280×720 pixels et d'un taux de rafraichissement de 30 images par seconde. La résolution a été fixée manuellement selon la taille maximale d'acquisition de données de profondeur possible avec la caméra utilisée.

Pour la moitié des plans, le robot a subi des repositionnements dans l'espace afin de simuler un déplacement et d'introduire du bruit dû à l'égo-mouvement de la plateforme robotique. Ce déplacement nous permet de confirmer l'efficience de l'approche lorsque le capteur reçoit des pertes de données visuelles dues à la vibration de la caméra lors d'un déplacement, une problématique importante pour un robot assistant qui doit souvent se déplacer et se repositionner.

Pour la phase d'expérimentation, on découpe le corpus en quatre différentes catégories, les deux premiers ensembles composés des plans de la première salle de test, avec et sans déplacement, et les deux autres ensembles composés de l'exécution des plans dans la seconde salle de test, avec ou sans mouvement. Il est aussi à noter que les plans de la catégorie 4 représentent des suites de plans et d'activités selon un taux de luminosité général différent du reste du corpus, en raison de l'enregistrement des vidéos de tests en soirée.

Lors de la phase d'entrainement du modèle, les jeux de données de la catégorie 1 ont été utilisés seulement pour l'entrainement et ceux des catégories 2-3 ont eu une extraction de 40 % de leurs images pour entrainer le classificateur. Les images extraites ont ensuite subi une étape d'étiquetage manuel de leurs objets selon leurs régions avant d'être ajoutées au jeu d'entrainement du classificateur d'objet. La catégorie 4, quant à elle, a été utilisée seulement lors de la phase d'évaluation pour le modèle de connaissance de base, afin de s'assurer d'éviter de causer un sur-apprentissage pour le module de classification et de fausser les résultats. Le choix d'utiliser des plans du corpus pour la phase d'entrainement et de validation en parallèle avec certaines images pour l'apprentissage des régions et des objets classifiables, a été fait afin d'assurer au classificateur de reconnaitre les différents environnements du corpus ainsi que ces critères de variances d'images.

Les critères de variances pour l'exécution des divers plans choisis sont les suivants :

— Variation de luminosité ;

— Variation de salles et ainsi d'arrière-plans ;

— Variation de personnes ;

— Variation d'angles visuels 2D pour les objets ;

— Variation d'occlusions par le placement d'objets l'un devant l'autre.

Afin d'entrainer le classificateur YOLO pour la détection et la classification d'objets, seulement 80 % des images ont reçu un étiquetage manuel de leurs régions avant d'être ajoutées au jeu d'entrainement.

Les régions des 20 % restantes ont été générées par l'algorithme 1, afin de confirmer sa pertinence. Dans la version actuelle du projet, ses régions générées sont sauvegardées localement dans un fichier externe avec une copie de l'image d'entrée, avant de subir une addition manuelle du nom des classes ainsi qu'un nettoyage des

classes d'objets d'arrière-plan détectées non utilisées par le Corpus. Ce processus va être remplacé à long terme par l'utilisation de services payants de classification, tel que l'api Google Cloud Vision[3], afin de permettre un fonctionnement dans PARC automatique plutôt que ce nécessiter des appels manuels externes.

Afin d'assurer un meilleur taux de reconnaissance des mains d'une personne lors de la phase de détection d'affordances, l'ensemble de données du corpus EgoHands[4] a été rajouté à l'ensemble du jeu d'entrainement du réseau de neurones. Celui-ci a aussi subi une phase additionnelle d'enrichissement afin d'améliorer ces caractéristiques. Pour ce faire, 70 % du jeu de données d'entrainement a été cloné et a subi des additions de divers bruits. Ce choix a été fait pour permettre au réseau YOLO un entrainement sur des caractéristiques plus variées qui n'existent pas dans la version actuelle des images du corpus. Ceci nous permet aussi d'éviter d'effectuer un sous-apprentissage pour la détection de régions dû à l'utilisation d'images non représentatives du corpus.

Les images ont été enrichies par diverses transformations géométriques, telles qu'une variation de rotation $\in]0, 360[$, des déplacements horizontal et vertical avec rembourrage par pixels noirs sur les zones maintenant vides et l'ajout de bruits gaussiens, sel et poivre et de chatoiement. Ces images bruitées permettent au classificateur d'apprendre des caractéristiques avec un plus haut taux d'invariance que celui disponible naturellement dans le corpus par rapport à la position des objets, de leurs angles et de leurs niveaux d'occlusion.

Deux modèles de classification, appelés **DrinkmakingBase** et **DrinkmakingAuto** ont été entrainés. Le premier modèle, **DrinkmakingBase**, est entrainé

3. `https://cloud.google.com/vision/`

4. http ://vision.soic.indiana.edu/projects/egohands/

seulement au jeu de données étiquetées manuellement et **DrinkmakingAuto** est entrainé au jeu d'images originales additionnées à celles dont les régions ont été générées par le module d'auto-apprentissage. Les deux modèles sont entrainés selon les paramètres suivants :

- Dimension d'image d'entrée : choix de re dimension alternante entre 288 × 288 et 384 × 384 pixels ;
- Rythme d'apprentissage : 0.001 ;
- Lot maximal : 500000 ;
- Politique d'entrainement : Par pas ;
- Pas : 400000, 450000 ;
- Ratio de recadrage d'images lors de la phase augmentation des données : 30 % ;
- Points d'ancrage : K-moyennes par métrique d'intersection sur union pour 7 clusters ;
- Niveau de confiance minimale pour la reconnaissance d'activité : 40 % (toute classification avec un seuil plus bas, même celles avec la bonne classe, sont ignorées).

On découpe les résultats de la phase d'expérimentation selon quatre catégories. Les deux premières sont les résultats de PARC sur les modèles **DrinkmakingBase** et **DrinkmakingAuto** avec une implémentation du modèle de réseau de neurones pour CPU grâce à la bibliothèque OpenCV. Les deux dernières sont sur les mêmes modèles, mais utilisent l'implémentation de réseau de neurones dans la bibliothèque d'origine, optimisée pour fonctionner sur GPU. On choisit de tester ainsi notre approche sur CPU et GPU afin de s'assurer que la bibliothèque CNN utilisée pour faire fonctionner le classificateur sur processeur nous offre des résultats comparables et ne fausse pas nos résultats.

5.2.1 Système de collecte de données

La conception de PARC est à sa base une approche multi-domaine afin de réaliser un module de reconnaissance pour robot assistant. Cette intégration en bas niveau de modules de reconnaissance d'objets, d'activités et de plans fait que l'analyse de l'efficience de l'approche nécessite un moyen d'analyser les différents modules séparément. C'est pourquoi on développe au sein de PARC un système de collecte de données personnalisé, qui, en plus d'enregistrer les informations essentielles, cible la vérification des modules de bas niveau par rapport à leur utilisation pour ceux de plus haut niveau.

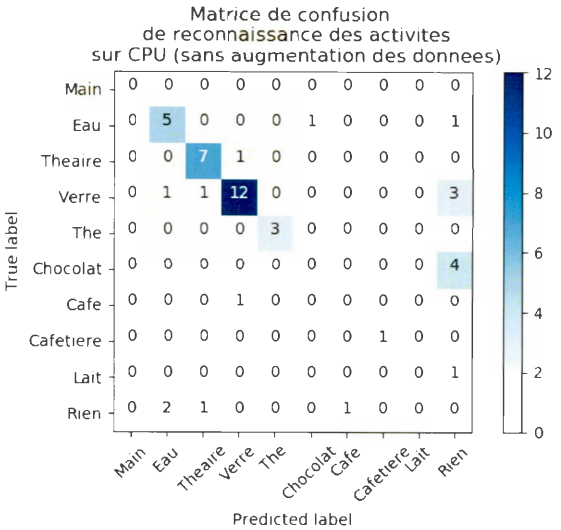
Dans un premier temps, on peut noter le résultat de classification des objets des vidéos pour chaque image. Ce jeu de données nous permet d'analyser l'apparition des catégories d'objets lors de l'exécution de la tâche de reconnaissance d'une vidéo. On peut alors utiliser ses données pour annoter quels objets ont été correctement classifiés ainsi que les régions d'une image classifiée incorrectement. Par incorrectement, nous voulons parler des régions avec objets incorrectement classifiés, ainsi que des régions retournées par le classificateur qui ne détiennent en réalité aucun objet.

On prend aussi en compte les activités reconnues par le système et leur moment d'apparition, regroupés selon celles qui sont envoyées au module de reconnaissance de plan et celles qui sont ignorées par ce système. Dans la version actuelle, cette démarcation se base sur une vérification que le taux de reconnaissance de l'activité est supérieur à un seuil de 80 %, c'est-à-dire que le système détecte une interaction entre la main et ledit objet dont le taux de classification est supérieur à 80 %, dépendamment si elles sont été détecté lors des deux dernières actions captées par le système. Ce choix a été fixé afin d'éviter de modéliser les actions parallèles, effectuées par deux mains différentes, telles que verser de l'eau d'une théière dans

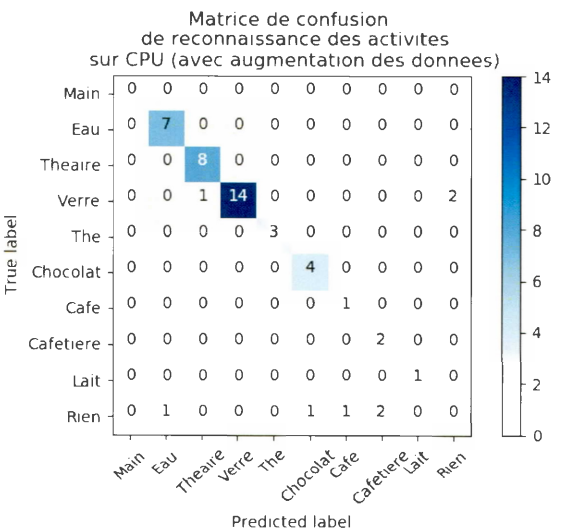
un verre. La dernière donnée acquise est le plan haut niveau retourné en sortie par le module de reconnaissance de plan, ainsi que ses prédictions d'actions futures.

5.2.2 Résultats

Dans le cas des matrices de confusion, on représente par une classe nommée "Rien" une classification ou une reconnaissance d'activité fantôme, c'est-à-dire qui n'existe pas réellement dans les données.

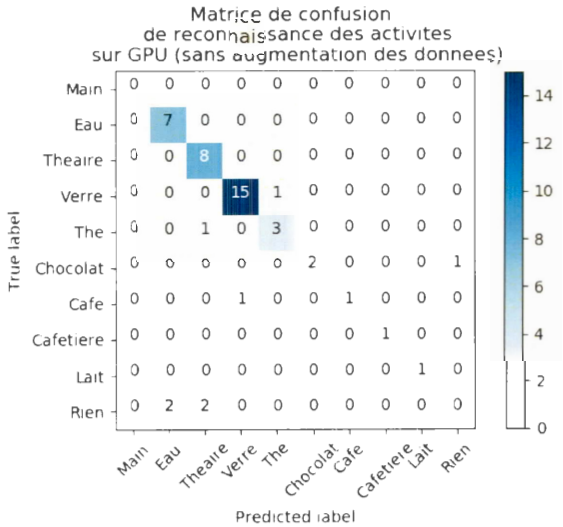


(a) ActCPU

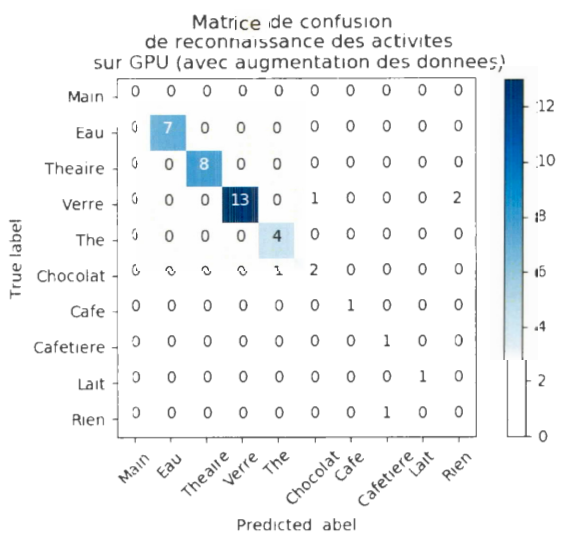


(b) ActCPUAug

Figure 5.3: Matrices de confusion du taux de reconnaissance d'activités, avec implémentation du classificateur sur CPU

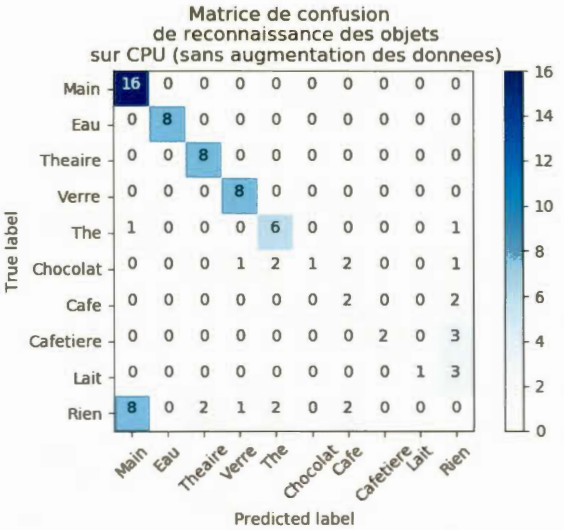


(a) ActGPU

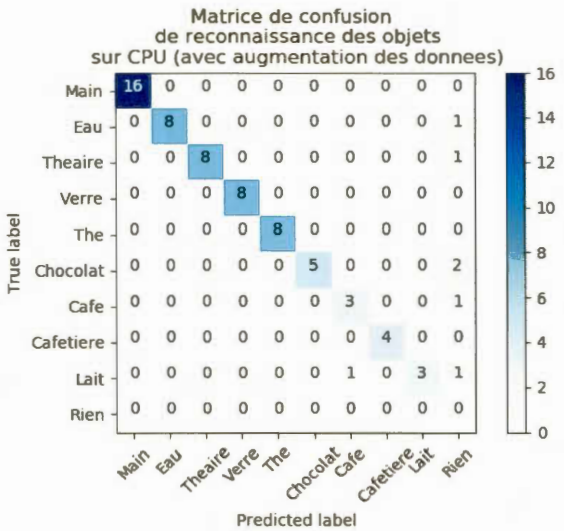


(b) ActGPUAug

Figure 5.4: Matrices de confusion du taux de reconnaissance d'activités, avec implémentation du classificateur sur GPU.

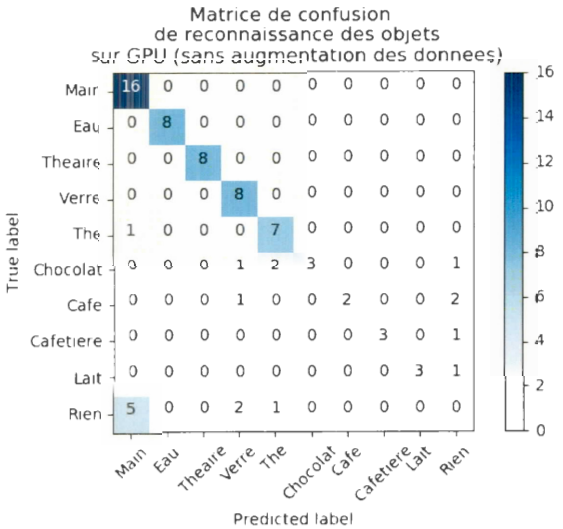


(a) ObjCPU

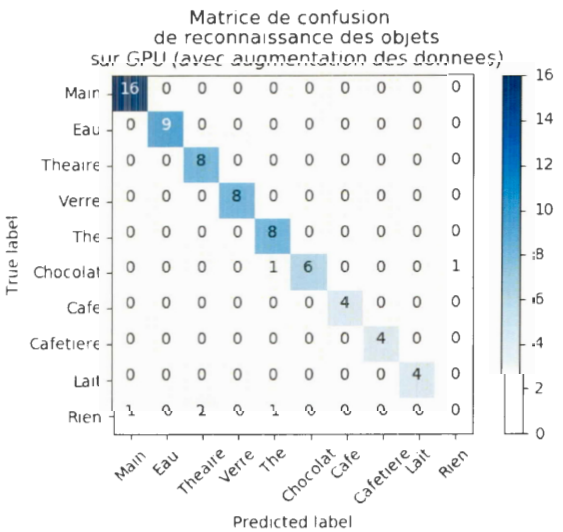


(b) ObjCPUAug

Figure 5.5: Matrices de confusion du taux de reconnaissance d'objets, avec implémentation du classificateur sur CPU.



(a) ObjGPU



(b) ObjGPUAug

Figure 5.6: Matrices de confusion du taux de reconnaissance d'objets, avec implémentation du classificateur sur GPU.

Tableau 5.1: Résultats de la reconnaissance d'activités

Reconnaissance d'activité	*Acc*	*Recall*	*Precision*	*F1*	*Fps*
CPU	0.61	0.61	0.63	0.62	10.48
CPU Augmenté	0.83	0.83	0.81	0.81	11.68
GPU	0.83	0.83	0.79	0.80	27.69
GPU Augmenté	0.88	0.88	0.92	0.88	28.89

Tableau 5.2: Résultats de la reconnaissance de plan

Reconnaissance de plan	**Acc**	*Rappel*	*Précision*	**F1**
CPU	0.5	0.5	0.875	0.58
CPU Augmenté	0.75	0.75	0.94	0.79
GPU	0.5	0.5	0.88	0.58
GPU Augmenté	0.88	0.98	0.91	0.88

5.3 Analyse globale des résultats

Les matrices de confusion 5.3, 5.4, 5.5 et 5.6 représentent les niveaux dè reconnaissance des activités et des objets de l'environnement dans le corpus. Les tableaux 5.1 et 5.2 représentent les mesures de performance du système pour la reconnaissance d'activités et de plans.

On peut noter un taux de précision de reconnaissance d'activités supérieur à 60 % même dans le cas minimal, atteignant 83 %. Peu importe la métrique utilisée, on confirme une augmentation de précision des activités reconnues et une diminution d'objets incorrectement classifiés lors de l'utilisation du module d'auto-apprentissage. On peut confirmer davantage l'efficience de l'auto-apprentissage par les matrices de confusion de la reconnaissance des objets. Plusieurs classifica-

tions fantômes de l'ensemble **DrinkmakingBase** disparaissent dans l'ensemble **DrinkmakingAuto**, peu importe l'utilisation d'un CPU ou GPU.

On peut expliquer cette amélioration significative des résultats par le fait que l'ajout de nouvelles régions d'entrainement aux connaissances du classificateur du robot lui a permis d'apprendre des caractéristiques précédemment inconnues au système, ainsi que de nouvelles régions possibles pour les situer.

On peut aussi remarquer qu'il existe une augmentation du taux de précision des métriques d'analyse lors de l'utilisation d'un GPU pour l'étape de classification d'objets, ainsi que de la vitesse de traitement. Après analyse, cette disparité provient du système du prétraitement de l'image d'entrée du classificateur pour CPU, qui nécessite des transformations additionnelles ne correspondant pas exactement au format d'entrée traditionnel du réseau de neurones. Cette différence peut aussi être expliquée par les différences en implémentation entre les deux bibliothèques. Toutefois, cette différence est amoindrie par le module d'auto-apprentissage, ce qui permet de conclure que le facteur primaire pour la précision du réseau de neurones est la richesse de son entrainement, et donc que l'approche peut fonctionner sur CPU et GPU. Elle nécessite seulement un bon jeu de données pour l'entrainer pour reconnaitre et fonctionner dans le foyer de la personne âgée que le module souhaite aider. Pour ce qui est du temps d'exécution, on peut confirmer que le réseau de neurones est le facteur limitant, puisque le reste du système est capable de fonctionner au même rythme que ses données en entrée. La vitesse de 11.68 images par seconde, quoique moins significative qu'espéré, est tout de même plus rapide que la vitesse normale d'un réseau de neurones sur processeur.

On a effectué une vérification des classifications incorrectes au niveau de la reconnaissance d'objets et d'activités afin de déceler les causes primaires des cas d'erreurs. Les cas d'échecs apparaissent principalement dans les vidéos de tests de

la catégorie 4, celles qui n'ont pas été utilisées lors de la phase d'entrainement. Après vérification manuelle des données, la cause principale d'échec provient de l'apparition de formats des classes d'objets uniques à ces vidéos. Par formats, on parle d'apparition des objets sous des angles non utilisés dans le reste des données, ainsi que l'apparition de lumières qui sont reflétées sur certaines catégories d'objets, telles que le chocolat, ce qui les a rendus non reconnaissables au système.

Un autre cas d'erreur, celui-ci au niveau de la reconnaissance d'activité a été décelé. Lors de la phase de recherche d'affordance, certains objets reconnus reçoivent dans une des vidéos de tests une occlusion significative des mains. Ces occlusions ne sont pas dues à une affordance main-objet, mais plutôt au placement de la main à quelques centimètres devant l'objet par rapport à la caméra. Lorsque l'occlusion cache une partie significative d'un objet, le calcul de profondeur de PARC est majoritairement basé sur les données de l'élément de premier plan, c'est-à-dire la main de l'être humain. Cette profondeur de main est ensuite utilisée dans le calcul d'affordance avec cette même main, ce qui amène à la découverte d'activités incorrectes.

5.4 Travaux futurs

Plusieurs pistes d'évolutions du système PARC existent, dont certaines sont actuellement en phase d'évolution.

Premièrement, les résultats montrent qu'il serait pertinent pour le module d'affordance d'effectuer une segmentation plus fine des éléments telle que l'utilisation d'algorithmes de détection de contours basée sur le centre des régions. La découverte des contours spécifiques des diverses classes d'objets plutôt que leurs zones de démarcations pourrait servir à calculer une distance de profondeur plus fine, ainsi que de localiser les zones affectées par l'affordance main-objet afin de confirmer que leurs données sont occluses.

Deuxièment, l'intégration du module avec les autres modules pertinents pour un robot assistant, tels que le module de modélisation des intérêts d'une personne, ou de son système de déplacement, ouvre plusieurs possibilités aux approches actuelles. Il serait alors possible d'ajouter aux modules de reconnaissance d'activités l'accès aux données de l'emploi du temps de l'individu, ses déplacements au courant de la journée, sa taille, etc., afin d'enrichir les observations fournies en entrées. Nous pourrions par exemple utiliser son module de détection de poses humaines, normalement utile pour détecter si la personne est tombée par terre, pour déceler l'angle et le positionnement de ses membres, plus particulièrement de ses bras et de son torse vis-à-vis des objets qu'il utilise. Cette info pourrait alors être utilisée pour inférer l'angle de l'objet qu'il utilise afin de le normaliser, ce qui permettrait de retirer entièrement le critère de variance d'angle lors de la phase de classification.

Troisièmement, une amélioration future utile serait la modélisation des affordances objets-objets afin de préciser des détails additionnels aux activités perçues, sans toutefois nécessiter un classificateur par parties. Nous pourrions par exemple détecter les affordances entre une fourchette et de la nourriture afin de modéliser l'activité *Manger* pour des plans tels que **Diner** ou **Souper**.

Une autre amélioration possible serait l'intégration des contraintes temporelles et physiques dans les bibliothèques de plans de PARC. Leur implémentation permettrait d'utiliser les données temporelles des activités acquises par PARC afin de raffiner les prédictions d'activités futures selon leur temps d'exécution et l'emplacement habituel où elles sont effectuées. Ces données permettraient ainsi de confirmer que les activités détectées d'un utilisateur sont effectuées selon une période de temps normal pour leur exécution et ainsi offrir un rappel si un oubli est perçu. Par exemple, PARC pourrait détecter que l'utilisateur a oublié d'effectuer l'activité **Fermer le poêle du four** après qu'un délai de 3 heures se soit

passé depuis l'activité **Ouvrir le poêle du four** un rappel qui peut être assez important à communiquer à la personne afin d'assurer sa sécurité. On pourrait aussi utiliser ces données pour aider le module de déplacement à se positionner aux endroits optimaux pour l'acquisition de données du capteur, puisqu'il serait possible de modéliser les plans et ses activités selon leurs salles, par exemple en limitant les plans *faire à déjeuner* aux positions à l'intérieur de la cuisine.

CONCLUSION

Dans ce mémoire, on a montré une intégration de différentes approches en vision par ordinateur, c'est-à-dire une classification d'objets et de reconnaissance d'activités pouvant s'implémenter à un module de reconnaissance de plan, et ce, dans le but de l'intégrer à un robot assistant cognitif. Ce robot permet d'offrir une aide cognitive à une personne âgé lors de ses activités de vies quotidienne. La conception d'un tel type d'agent nécessite des solutions efficientes pour reconnaitre les actions et intentions de son propriétaire. L'objectif primaire du mémoire est d'offrir un module de reconnaissances d'actions, intégrable à un système de reconnaissance de plan, d'un utilisateur capable de s'exécuter en temps réel sur une machine locale seulement.

Son approche consiste à utiliser les données d'un capteur visuel sous format d'image couleur et de profondeur, et d'extraire les activités en cours d'exécution ainsi que leur durée. Ce système de reconnaissance utilise des techniques d'intelligence artificielle, d'apprentissage machine et de psychologie afin de découvrir les interactions humain-objet, les intentions derrière ces gestes ainsi que pour prédire les actions futures afin de mieux les détecter.

Pour ce faire, un système de reconnaissance d'activités est implémenté, ainsi qu'un nouveau modèle de réseaux de neurones convolutifs basée sur DarknetV3Tiny et un nouvel algorithme de génération de régions, nommé recherche sélective par profondeur.

Le système a été conçu en trois temps. Dans un premier temps, un module de reconnaissance d'objet est implémenté selon les techniques recensées dans l'état de

l'art. Le module est découpé en deux structures parallèles. La première a pour but d'effectuer une reconnaissance et une classification de formes dans une image couleur, afin de détecter l'emplacement probable d'objets dans l'environnement. Pour ce faire, un modèle de réseau neuronal convolutif avec découpe de régions basé sur un modèle YOLOV3 est conceptualisée, entraînée et déployée pour permettre d'atteindre un juste milieu entre le niveau de précision et la vitesse d'exécution. Le réseau est optimisé pour offrir de bonnes performances de classification lorsqu'il roule, soit sous un processeur, soit sous une carte graphique. La seconde technique propose un algorithme étendu des techniques de l'état de l'art en regroupement de régions, afin d'assurer l'amélioration continue du processus de détection. Dans les cas où les connaissances du robot assistant comportent des lacunes lors de son intégration dans un nouveau milieu, ce mécanisme effectue une seconde passe pour déceler les informations visuelles qui auraient été manquées par le système central. Ces informations sont alors ajoutées à son module de classification, qui peut les utiliser pour mettre ses connaissances à jour sur les caractéristiques importantes à détecter lors de la phase de détection d'objets.

Dans un second temps, le module utilise les caractéristiques visuelles extraites par l'approche, afin de déceler les actions d'un être humain en cours d'exécution devant le capteur. La méthode de reconnaissance utilise le principe des affordances pour détecter les interactions entre les mains d'une personne et un objet, afin de modéliser les actions en cours et leur durée. Dans un dernier temps, le système communique avec un module de reconnaissance de plan pour l'avertir de l'action en cours. Ce module utilise alors cette donnée pour inférer l'intention primaire de l'utilisateur selon la séquence d'actions passées et transmet les intentions de cet utilisateur au système pour étendre et raffiner les activités futures à reconnaitre.

L'approche proposée dans ce mémoire a servi de base pour la création du module PARC, **Plan and Activity Recognition Component**, pour le réseau AGE-

WELL. Ce module a été testé en contexte réel grâce aux jeux de tests décrits au chapitre 5. Pour ce faire, un corpus a été conçu pour représenter des activités de la vie de tout les jours, tels que la préparation d'un café le matin ou la préparation d'une tisane. On a ensuite testé le module sur ce corpus selon différentes configurations, afin de confirmer la viabilité de l'approche ainsi que pour s'assurer qu'elle offre des taux de précisions élevés sous différentes configurations de système d'assistant robotique, telles que la possibilité d'accéder à une carte graphique ou à un processeur lors de l'étape de la détection des objets dans le champ visuel du capteur. Les résultats de cette expérience montrent la force du module pour la détection d'activités en cours, grâce à ces observations prises d'images. Ils démontrent également une forte capacité d'amélioration de son taux de précision à long terme, grâce à son système d'auto-apprentissage, qui lui permet d'ajouter des représentations de ces diverses classes d'objets à détecter sur de nouvelles formes et avec différents facteurs de variance.

Les algorithmes utilisés dans ce mémoire sont variés, mais reposent tous sur une modélisation partielle de l'environnement. Toutefois, ce modèle est loin d'être parfait et peut être enrichi par divers moyens à long terme. L'intégration des contraintes temporelles des plans et de leurs activités respectives permettrait au module de reconnaitre les oublis d'activités d'un utilisateur dans une séquence d'actions temporelle. Finalement, l'intégration du module sur la plateforme robotique complète permettra d'utiliser les informations de divers modules, tels que la reconnaissance de l'emploi du jour et les déplacements du robot. Ces données contiennent des informations pertinentes sur certaines activités, par exemple l'emplacement dans la maison de l'utilisateur, ce qui améliorerait le taux de précision du module et lui permettrait de repositionner son capteur pour maximiser ses chances de capter une activité en cours d'exécution.

APPENDICE A

CODE SOURCE

Le code source du projet PARC peut être retrouvé partiellement au lien suivant :

`https://github.com/MathGravel/ImageRec`

RÉFÉRENCES

Adelson, E. H., Anderson, C. H., Bergen, J. R., Burt, P. J. et Ogden, J. M. (1984). Pyramid methods in image processing. *RCA engineer*, *29*(6), 33–41.

Alexe, B., Deselaers, T. et Ferrari, V. (2010). What is an object ? Dans *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, 73–80. IEEE.

Andreadis, S., Stavropoulos, T. G., Meditskos, G. et Kompatsiaris, I. (2016). Dem@ home : Ambient intelligence for clinical support of people living with dementia. Dans *International Semantic Web Conference*, 357–368. Springer.

Arbeláez, P., Pont-Tuset, J., Barron, J., Marques, F. et Malik, J. (2014). Multiscale Combinatorial Grouping. Dans *Computer Vision and Pattern Recognition*, p. s.p.

Arcelus, A., Jones, M. H., Goubran, R. et Knoefel, F. (2007). Integration of Smart Home Technologies in a Health Monitoring System for the Elderly. Dans *Advanced Information Networking and Applications Workshops, 2007, AINAW'07. 21st International Conference on*, volume 2, 820–825. IEEE.

Bay, H., Ess, A., Tuytelaars, T. et Van Gool, L. (2008). Speeded-up robust features (SURF). *Computer vision and image understanding*, *110*(3), 346–359.

Bengio, Y. *et al.* (2009). Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, *2*(1), 1–127.

Blanc-Durand, P. (2018). *Réseaux de neurones convolutifs en médecine nucléaire : Applications à la segmentation automatique des tumeurs gliales et à la correction d'atténuation en TEP/IRM.* (Thèse de doctorat). Paris Decartes. `http://dx.doi.org/10.13140/RG.2.2.23231.97441`

Blasco, R., Marco, Á., Casas, R., Cirujano, D. et Picking, R. (2014). A smart kitchen for ambient assisted living. *Sensors*, *14*(1). 1629–1653.

Broekens, J., Heerink, M., Rosendal, H. *et al.* (2009). Assistive social robots in elderly care : A review. *Gerontechnology*, *8*(2), 94–103.

Canada, S. (2012). *Recensement Canadien par sexe et âge.* www.statcan.gc.ca/pub/91-215-x/2012000/part-partie2-eng.htm.

Canny, J. (1987). A computational approach to edge detection. In *Readings in Computer Vision* 184–203. Elsevier.

Cavanagh, P. (2011). Visual cognition. *Vision research*, *51*(13), 1538–1551.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K. et Yuille, A. L. (2017). Deeplab : Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, *40*(4), 834–848.

Choodarathnakara, A., Kumar, T. A., Koliwad, S. et Patil, C. (2012). Mixed pixels : a challenge in remote sensing data classification for improving performance. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, *1*(9), pp–261.

Dutta, V. et Zielinska, T. (2017). Action prediction based on physically grounded object affordances in human-object interactions. Dans *Robot Motion and Control (RoMoCo), 2017 11th International Workshop on*, 47–52. IEEE.

Feil-Seifer, D. et Mataric, M. J. (2005). Defining socially assistive robotics. Dans *Rehabilitation Robotics, 2005. ICORR 2005. 9th International Conference on*, 465–468. IEEE.

Felzenszwalb, P. F. et Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *International journal of computer vision*, *59*(2), 167–181.

F.Felzenszwalb, R.B.Girshick, D. e. D.-m. (2010). Object detection with discriminatively trained partbased models. Dans *IEEE Transactions on Pattern Analysis andMachine Intelligence*, 1627–1645. IEEE.

Fougère, M., Harvey, S., Mercenier, J. et Mérette, M. (2009). Population ageing, time allocation and human capital : A general equilibrium analysis for Canada. *Economic Modelling*, *26*(1), 30–39.

Geib, C. W. et Goldman, R. P. (2009). A probabilistic plan recognition algorithm based on plan tree grammars. *Artificial Intelligence*, *173*(11), 1101–1132.

Gibson, J. J. (1966). *The senses considered as perceptual systems.* Houghton Mifflin.

Gibson, J. J. (1979). *The Ecological Approach to Visual Perception.* Houghton Mifflin.

Gidaris, S. et Komodakis, N. (2015). Object detection via a multi-region and

semantic segmentation-aware CNN model. Dans *Proceedings of the IEEE International Conference on Computer Vision*, 1134–1142.

Girshick, R. (2015). Fast R-CNN. Dans *Proceedings of the IEEE international conference on computer vision*, 1440–1448.

Girshick, R., Donahue, J., Darrell, T. et Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. Dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, 580–587.

Gregory, R. L. (1970). The intelligent eye. *American Journal of Optometry and Archives of American Academy of Optometry*, *48*(10), 851.

Grezes, J. et Decety, J. (2002). Does visual perception of object afford action? Evidence from a neuroimaging study. *Neuropsychologia*, *40*(2), 212–222.

Gross, H.-M., Mueller, S., Schroeter, C., Volkhardt, M., Scheidig, A., Debes, K., Richter, K. et Doering, N. (2015). Robot companion for domestic health assistance : Implementation, test and case study under everyday conditions in private apartments. Dans *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, 5992–5999. IEEE.

Gross, H.-M., Schroeter, C., Mueller, S., Volkhardt, M., Einhorn, E., Bley, A., Martin, C., Langner, T. et Merten, M. (2011). Progress in developing a socially assistive mobile home robot companion for the elderly with mild cognitive impairment. Dans *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, 2430–2437. IEEE.

Harris, C. et Stephens, M. (1988). A combined corner and edge detector. Dans *Alvey vision conference*, volume 15, 10–5244. Citeseer.

He, K., Gkioxari, G., Dollár, P. et Girshick, R. (2017). Mask R-CNN. Dans *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2980–2988. IEEE.

He, K., Zhang, X., Ren, S. et Sun, J. (2016). Deep residual learning for image recognition. Dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hosang, J., Benenson, R., Dollár, P. et Schiele, B. (2016). What makes for effective detection proposals? *IEEE transactions on pattern analysis and machine intelligence*, *38*(4), 814–830.

Hosang, J., Benenson, R. et Schiele, B. (2014). How good are detection proposals, really? *arXiv preprint arXiv :1406.6962*.

Hutson, S., Lim, S. L., Bentley, P. J., Bianchi-Berthouze, N. et Bowling, A. (2011). Investigating the suitability of social robots for the wellbeing of the elderly. Dans *International Conference on Affective Computing and Intelligent Interaction*, 578–587. Springer.

Ikemura, S. et Fujiyoshi, H. (2010). Real-time human detection using relational depth similarity features. Dans *Asian Conference on Computer Vision*, 25–38. Springer.

Kanellopoulos, I. et Wilkinson, G. G. (1997). Strategies and best practice for neural network image classification. *International Journal of Remote Sensing*, *18*(4), 711–725.

Katiyar, S. et Arun, P. (2014). Comparative analysis of common edge detection techniques in context of object extraction. *arXiv preprint arXiv :1405.6132*.

Koppula, H. et Saxena, A. (2013a). Learning spatio-temporal structure from RGB-D videos for human activity detection and anticipation. Dans *International Conference on Machine Learning*, 792–800.

Koppula, H. S., Gupta, R. et Saxena, A. (2013). Learning human activities and object affordances from RGB-D videos. *The International Journal of Robotics Research*, *32*(8), 951–970.

Koppula, H. S. et Saxena, A. (2013b). Anticipating human activities for reactive robotic response. Dans *IROS*, p. 2071. Tokyo.

Kostavelis, I., Giakoumis, D., Malasiotis, S. et Tzovaras, D. (2015). RAMCIP : towards a robotic assistant to support elderly with mild cognitive impairments at home. Dans *International Symposium on Pervasive Computing Paradigms for Mental Health*, 186–195. Springer.

Krishnan, N. C. et Cook, D. J. (2014). Activity recognition on streaming sensor data. *Pervasive and mobile computing*, *10*, 138–154.

Krizhevsky, A., Sutskever, I. et Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Dans *Advances in neural information processing systems*, 1097–1105.

Lampert, C. H., Blaschko, M. B. et Hofmann, T. (2009). Efficient subwindow search : A branch and bound framework for object localization. *IEEE transactions on pattern analysis and machine intelligence*, *31*(12), 2129–2142.

Le, Q. V. (2013). Building high-level features using large scale unsupervised learning. Dans *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 8595–8598. IEEE.

LeCun, Y., Bengio, Y. et Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436.

Liciotti, D., Frontoni, E., Zingaretti, P., Bellotto, N. et Duckett, T. (2017). HMM-based activity recognition with a ceiling RGB-D Camera. Dans *ICPRAM*, p. s.p. s.l.

Lin, T.-Y., Goyal, P., Girshick, R., He, K. et Dollár, P. (2018). Focal loss for dense object detection. *IEEE transactions on pattern analysis and machine intelligence*.

Liu, C., Sharan, L., Adelson, E. H. et Rosenholtz, R. (2010). Exploring features in a bayesian framework for material recognition. Dans *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, 239–246. IEEE.

Lopes, M., Melo, F. S. et Montesano, L. (2007). Affordance-based imitation learning in robots. Dans *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, 1015–1021. IEEE.

Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. *International journal of computer vision*, *60*(2), 91–110.

Maas, A. L., Hannun, A. Y. et Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. Dans *Proc. icml*, volume 30, p. 3.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. et Riedmiller, M. (2013). Playing Atari with deep reinforcement learning. *arXiv preprint arXiv :1312.5602*.

Paez, A., Mercado, R. G., Farber, S., Morency, C. et Roorda, M. (2010). Accessibility to health care facilities in Montreal Island : an application of relative accessibility indicators from the perspective of senior and non-senior residents. *International journal of health geographics*, *9*(1), 52.

Pollack, M. E., Brown, L., Colbry, D., McCarthy, C. E., Orosz, C., Peintner, B., Ramakrishnan, S. et Tsamardinos, I. (2003). Autominder : An intelligent cognitive orthotic system for people with memory impairment. *Robotics and Autonomous Systems*, *44*(3-4), 273–282.

Pollack, M. E., Brown, L., Colbry, D., Orosz, C., Peintner, B., Ramakrishnan, S., Engberg, S., Matthews, J. T., Dunbar-Jacob, J., McCarthy, C. E. *et al.* (2002). Pearl : A mobile robotic assistant for the elderly. Dans *AAAI workshop on automation as eldercare*, volume 2002, 85–91.

Portilla, J., Strela, V., Wainwright, M. J. et Simoncelli, E. P. (2003). Image

denoising using scale mixtures of Gaussians in the wavelet domain. *IEEE Transactions on Image processing*, *12*(11), 1338–1351.

Redmon, J., Divvala, S., Girshick, R. et Farhadi, A. (2016). You only look once : Unified, real-time object detection. Dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, 779–788.

Redmon, J. et Farhadi, A. (2017). YOLO9000 : Better, Faster, Stronger. Dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7263–7271.

Redmon, J. et Farhadi, A. (2018). Yolov3 : An incremental improvement. *arXiv preprint arXiv :1804.02767*.

Ren, S., He, K., Girshick, R. et Sun, J. (2015). Faster R-CNN : Towards real-time object detection with region proposal networks. Dans *Advances in neural information processing systems*, 91–99.

Robinson, H., MacDonald, B. et Broadbent, E. (2014). The role of healthcare robots for older people at home : A review. *International Journal of Social Robotics*, *6*(4), 575–591.

Rublee, E., Rabaud, V., Konolige, K. et Bradski, G. (2011). ORB : An efficient alternative to SIFT or SURF. Dans *Computer Vision (ICCV), 2011 IEEE international conference on*, 2564–2571. IEEE.

Russell, S. J. et Norvig, P. (2016). *Artificial Intelligence : A Modern Approach*. Malaysia ; Pearson Education Limited,.

Rybok, L. (2017). *Unsupervised object candidate discovery for activity recognition*. (Thèse de doctorat). Karlsruher Instituts für Technologie.

Schroeter, C., Mueller, S., Volkhardt, M., Einhorn, E., Huijnen, C., van den Heuvel, H., van Berlo, A., Bley, A. et Gross, H.-M. (2013). Realization and user evaluation of a companion robot for people with mild cognitive impairments. Dans *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, 1153–1159. IEEE.

Sung, J., Ponce, C., Selman, B. et Saxena, A. (2011). Human Activity Detection from RGBD images. *plan, activity, and intent recognition*, *64*.

Sural, S., Qian, G. et Pramanik, S. (2002). Segmentation and histogram generation using the HSV color space for image retrieval. Dans *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 2, II–II. IEEE.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D.,

Vanhoucke, V. et Rabinovich, A. (2015). Going deeper with convolutions. Dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1–9.

Uijlings, J. R., Van De Sande, K. E., Gevers, T. et Smeulders, A. W. (2013). Selective Search for Object Recognition. *International journal of computer vision*, *104*(2), 154–171.

Van de Weijer, J., Gevers, T. et Bagdanov, A. D. (2006). Boosting color saliency in image feature detection. *IEEE transactions on pattern analysis and machine intelligence*, *28*(1), 150–156.

Wang, X. et Wang, Y. (2014). Improving content-based and hybrid music recommendation using deep learning. Dans *Proceedings of the 22nd ACM international conference on Multimedia*, 627–636. ACM.

Wördenweber, B., Boyce, P., Hoffman, D. D. et Wallaschek, J. (2007). *Automotive lighting and human vision*. Springer.

Xia, L. et Aggarwal, J. (2013). Spatio-temporal depth cuboid similarity feature for activity recognition using depth camera. Dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2834–2841.

Yujian, L. et Bo, L. (2007). A normalized levenshtein distance metric. *IEEE transactions on pattern analysis and machine intelligence*, *29*(6), 1091–1095.

Zeiler, M. D. et Fergus, R. (2014). Visualizing and understanding convolutional networks. Dans *European conference on computer vision*, 818–833. Springer.

Zhang, K., Zuo, W., Chen, Y., Meng, D. et Zhang, L. (2017). Beyond a gaussian denoiser : Residual learning of deep CNN for image denoising. *IEEE Transactions on Image Processing*, *26*(7), 3142–3155.

Zhang, Z. (2012). Microsoft Kinect sensor and its effect. *IEEE multimedia*, *19*(2), 4–10.

Zheng, L., Yang, Y. et Tian, Q. (2017). SIFT meets CNN : A decade survey of instance retrieval. *IEEE transactions on pattern analysis and machine intelligence*.

Zitnick, C. L. et Dollár, P. (2014). Edge boxes : Locating object proposals from edges. Dans *European Conference on Computer Vision*, 391–405. Springer.