Reconnaissance d'actions humaines dans des vidéos, en particulier lors d'interaction avec des objets

Camille Maurice

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ FÉDÉRALE TOULOUSE MIDI-PYRÉNÉES

Délivré par :
l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)

Présentée et soutenue le *10/12/2020* **par :**

CAMILLE MAURICE

Reconnaissance d'actions humaines dans des vidéos, en particulier lors d'interaction avec des objets

---

JURY

JENNY BENOIS-PINEAU	Professeure des Universités	Rapporteure
MOHAMED DAOUDI	Professeur	Examinateur
SÉVERINE DUBUISSON	Maître de conférences	Rapporteure
PHILIPPE JOLY	Professeur des Universités	Président du Jury
FRÉDÉRIC LERASLE	Professeur des Universités	Directeur de thèse

---

École doctorale et spécialité :
EDSYS : Robotique 4200046
Unité de Recherche :
Laboratoire d'analyse et d'architecture des systèmes
Directeur de Thèse :
Frédéric LERASLE
Rapporteurs :
Jenny BENOIS-PINEAU et *Séverine DUBUISSON*

Résumé :

Dans cette thèse nous étudions la reconnaissance d'actions humaines. Typiquement, différentes actions se déroulent dans un même lieu et font intervenir divers objets. Ce problème est difficile en raison de la variété et la ressemblance de certaines actions, de l'encombrement du fond de la scène. De nombreuses approches de vision par ordinateur étudient cette problématique et leur performance est souvent dépendante du paramétrage de certains hyper-paramètres. Par exemple pour les approches d'apprentissage profond nous retrouvons l'initialisation du learning-rate, la taille des mini-lots... Partant de ce constat, nous commençons par une étude comparative des outils d'optimisation des hyper-paramètres de la littérature appliquée à une problématique de vision par ordinateur. Puis nous proposons une première approche bayésienne originale pour la reconnaissance d'actions en ligne qui repose sur des primitives de haut-niveau en 3D : l'observation du squelette humain et les objets environnants. Les nombreux paramètres à régler sont optimisés grâce à l'outil d'optimisation qui émerge de notre étude comparative. Les performances de cette première approche sont comparées à un réseau d'apprentissage profond de l'état de l'art, il en ressort une certaine complémentarité que nous proposons d'exploiter à travers un mécanisme de fusion. Enfin, suite aux récentes avancées dans les réseaux de convolutions à graphes, nous proposons une approche compacte originale et modulaire qui repose sur la construction de graphes spatio-temporels du squelette et des objets. Ces différentes approches sont évaluées et comparées, en performance brute et vis-à-vis des actions sous-représentées sur différents jeux de données publiques qui proposent des séquences d'actions de la vie quotidienne. Nos approches montrent des gains de performance intéressants eu égard à la littérature, notamment vis-à-vis des classes sous représentées dans le jeu de données.

Mots clés : vision par ordinateur, reconnaissance d'actions, interactions homme objet

Abstract :

In this thesis we study the recognition of actions of daily life. Typically, different actions take place in the same place and involve various objects. This problem is difficult because of the variety and resemblance of some actions and the clutter in the background. Many computer vision approaches study this problem and their performance is often dependent on the setting of certain hyper-parameters. For example, for deep learning approaches there are : the initialization of the learning-rate, the size of the mini-batch... Based on this observation, we begin with a comparative study of hyper-parameter optimization tools from the literature applied to a computer vision problem. Then we propose a first Bayesian approach for online action recognition based on high-level 3D primitives : the observation of the human skeleton and surrounding objects. The parameters to be set are optimized thanks to the optimization tool that emerges from our comparative study. The performances of this first approach are compared to a deep state of the art learning network, and a certain complementarity emerges that we propose to exploit through a fusion mechanism. Finally, following recent advances in graph convolution networks, we propose a light and modular approach based on the construction of spatio-temporal graphs of the skeleton and objects. The validity of the different approaches is evaluated, in raw performance and with respect to under-represented actions on different public data sets that propose sequences of actions of everyday life. Our approaches show intereting results compared to the litterature especially regarding imbalanced data and under-represented classes in datasets.

Keywords : computer vision, action recognition, human object interaction

Remerciements

Je souhaite tout d'abord adresser ces remerciements à Jenny Benois-Pineau et Séverine Dubuisson pour avoir accepté d'être les rapporteures de ma thèse ainsi que pour la pertinence de leurs remarques sur le manuscrit. Je remercie également Mohamed Daoudi et Philippe Joly pour avoir participé au jury et pour les échanges enrichissants lors de la soutenance de ma thèse.

Ces travaux de thèse ont été menés sous la direction de Frédéric Lerasle au sein du Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS) à Toulouse. Je tiens à le remercier car grâce à ses précieux conseils, sa disponibilité et son professionnalisme, il m'a permis de mener à bien mes travaux. L'exigence dont il a su faire preuve avec mesure, ainsi que son implication dans mes travaux m'ont permis de me dépasser et de ressortir grandie de cette expérience.

Je souhaite également adresser quelques mots à l'attention de l'équipe RAP et de son directeur Patrick Danès. Je tiens à les remercier au complet pour m'avoir fait confiance et accepté le financement de la thèse. Je me suis sentie pleinement intégrée dans l'équipe et ce dès mon arrivée au LAAS en tant qu'ingénieure de recherche. Grâce à eux, ces quatre années ont été à la fois une incroyable expérience professionnelle et humaine.

Tout ce chemin n'aurait pas été possible si je n'avais pas suivi le Master IPCV et je tiens à remercier Aurélie Bugeau pour son implication et le suivi qu'elle nous a accordé durant ces deux années riches en aventures. Plus généralement, je remercie les personnes qui m'ont fait confiance durant ces années d'étude. Je remercie un précédent maître de stage, Franck Mamalet, qui m'a permis de rencontrer Frédéric Lerasle. Également une mention spéciale pour l'équipe de la fondation La Dépêche.

Ces dernières années ont été très intenses et riches en émotions, je remercie mes collègues et je garde en mémoire nos innombrables repas à la cafétéria. Je tiens à remercier mes amis du Master IPCV, Antoine, Charlotte, Pierre et Philippe qui ont été d'un grand soutien et source de grands fous rires. Mes amis à Toulouse qui m'ont permis de garder une vie sociale : Aurélien, Ingrid, Michalina, Stefen, Tony, Xavier... Mes amis à Bordeaux que j'ai moins vus : Adrien, Loic, Pierre... Je n'oublie pas François qui m'a également beaucoup aidée.

Je remercie également mon compagnon Frédéric pour son soutien durant déjà 9 années ainsi que pour son implication dans la logistique de la vie quotidienne durant l'intense rédaction. Je n'oublie pas ses relectures minutieuses à la recherche de fautes.

Enfin, j'adresse un ultime remerciement à mes parents : Isabelle et Yannick qui ont toujours accordé une grande valeur aux études et à la persévérance. Une qualité qui m'a été essentielle pour réussir à soutenir ma thèse.

Liste des publications

Journal international

1. Francisco Madrigal, Camille Maurice et Frédéric Lerasle. Hyper-parameter optimization tools comparison for Multiple Object Tracking applications. *Machine Vision and Applications*, Springer, 2019, 30.2, pp.269-289.

Conférences internationales

1. Camille Maurice, Francisco Madrigal et Frédéric Lerasle. Hyper-Optimization tools comparison for parameter tuning applications. *14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS 2017)*, Août 2017, Lecce, Italie.

2. Camille Maurice, Francisco Madrigal, André Monin et Frédéric Lerasle. A New Bayesian Modeling for 3D Human-Object Action Recognition. *16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS 2019)*, Sep. 2019, Taipei, Taiwan.

3. Camille Maurice, Francisco Madrigal et Frédéric Lerasle. Late Fusion of Bayesian and Convolutional Models for Action Recognition. *25th IEEE International Conference on Pattern Recognition (ICPR 2020)*, Jan. 2021, Milan (en ligne), Italie.

4. (soumission en cours) Camille Maurice, Francisco Madrigal et Frédéric Lerasle. Fusion of Multiple Graph Convolutional Networks for Action Recognition. *Winter Conference on Applications of Computer Vision*, IEEE WACV 2021.

Conférence nationale

1. Camille Maurice, Francisco Madrigal et Frédéric Lerasle. Fusion de modèles bayésiens et de convolution pour la reconnaissance d'actions. *Reconnaissance des Formes, Image, Apprentissage et Perception (RFIAP 2020)*, Jun. 2020, Vannes (en ligne), France.

Table des matières

Liste des abréviations

ANBM	**A** **N**ew **B**ayesian **M**odel, notre modèle présenté dans le chapitre 3
ANBM-C3D-GRU	Fusion tardive prédictions des approches ANBM et C3D-GRU
CNN	**C**onvolutional **N**eural **N**etwork
CRF	**C**onditional **R**andom **F**ield
GNN	**G**raph **N**eural **N**etwork
GRU	**G**ated **R**eccurent **U**nit
HMM	**H**idden **M**arkov **M**odel
LSTM	**L**ong **S**hort **T**erm **M**emory
MoCap	**Mo**tion **Cap**ture
MOT	**M**ultiple **O**bject **T**racking, suivi multi-objects
MOTA	**M**ultiple **O**bject **T**racking **A**ccuracy
RNN	**R**ecurrent **N**eural **N**etwork
GCN	**R**éseaux de **C**onvolutions à **G**raphes
SKLOGCN	**SK**e**L**eton **O**bject **G**raph **C**onvolutional **N**etwork
STGCN	**S**patio-Temporal **G**raph **C**onvolution **N**etwork

Liste des tableaux

Table des figures

CHAPITRE 1

Introduction

Dans cette thèse nous nous intéressons à la reconnaissance visuelle d'actions exécutées par une personne. En particulier au travers de l'observation conjointe de la posture et des objets de la scène avec lesquels il est possible d'interagir.

1.1 Motivations

Données vidéos et applications

Les capteurs vidéo sont très largement répandus au sein de la population. Avec l'utilisation des smartphones ou des caméras d'actions, il est courant d'enregistrer des vidéos. Dans le domaine de la surveillance de la voie publique, les capteurs vidéos se multiplient également, par exemple à Toulouse, le nombre de caméras dans l'espace publique est passé de 21 à 400 appareils entre 2014 et 2020 [lad 2020]. En conséquence, le nombre d'heures de vidéo enregistrées est considérable et les services d'hébergement et de diffusion sur le web tels que YouTube s'adaptent pour gérer un tel afflux. En effet en 2020, un milliard d'heures sont regardées par jour [you 2020]. Un tel essor dans la quantité et la disponibilité de vidéos contribue à l'émergence de nouveaux besoins dans l'analyse automatique de vidéos et de nouvelles applications associées.

Dans le domaine de la vidéo-surveillance, la communauté Vision s'est intéressée à l'analyse automatique sur des vidéos dès 2007 avec le jeu de données PETS [Ferryman 2007]. Le but étant de reconnaître un comportement suspect ou des bagages abandonnés. Cela facilite le travail de recherche d'une séquence particulière parmi des enregistrements continus qui peuvent s'étaler sur des semaines. Cette application est illustrée par la figure 1.1b.

La reconnaissance automatique d'actions a également des applications dans le domaine de l'indexation vidéo. Par exemple dans le développement de moteurs de recherche vidéo afin de classer et retrouver une vidéo à partir de mots-clés ou pour l'organisation de collections de vidéos personnelles. Elle peut aussi servir à l'amélioration des performances des sportifs grâce à une analyse fine de leur mouvements et de leurs actions [Martin 2019]. En effet il est ainsi possible pour le coach d'analyser quantitativement les techniques du sportif et de ses adversaires pour améliorer son programme d'entraînement.

Dans la robotique, il existe un domaine proche de notre problématique de reconnaissance d'actions à partir de l'observation conjointe de l'homme et des objets : la cobotique. M.Peshkin et J.E.Colgate [Peshkin 1999] sont les premiers à introduire le terme de cobot. Ils le définissent comme un robot conçu pour une interaction directe avec un opérateur humain, dans un espace

(a) Cobotique - Technologie de captation des gestes humains (Mines Paris Tech - PSA Peugeot Citroën)

(b) Jeu de données de vidéo-surveillance [Lien 2020]

FIGURE 1.1 – Applications

de travail partagé. À l'origine cela n'impliquait pas nécessairement de capteur visuel car les hommes et les robots partageaient un même espace de travail mais l'enjeu était surtout au niveau de la sécurité et de l'utilisation de capteurs de force. Maintenant les nouveaux cas d'utilisation incluent des situations de collaboration plus avancées où le robot et l'opérateur interagissent certes dans un espace de travail partagé mais en plus sur une même pièce par exemple. D'où la nécessité d'ajouter des capteurs visuels et des systèmes de reconnaissance d'actions pour la communication et collaboration entre les deux parties. De tels cobots peuvent être déployés dans un cadre domestique pour l'assistance aux personnes ou dans un cadre industriel comme illustré par la figure 1.1a.

Les données vidéos sont largement répandues et les avancées dans l'analyse vidéo permettent d'envisager de nombreuses applications. En particulier il existe de nombreux détecteurs de posture ou d'objets que nous abordons brièvement dans la section suivante.

Reconnaissance de la posture et des objets

Cette quantité de données images ou vidéos grandissante et disponibles sur le web a permis la création de larges jeux de données. En effet au lieu d'enregistrer les données nécessaires il est possible de collecter les URL des images existantes. Les images peuvent être recherchées par des mots-clés sur Google ou Flickr, un site d'hébergement de photos prises par des photographes amateurs. Quand aux données vidéos il est de plus en plus populaire de les collecter à partir de films ou de YouTube. Les jeux de données ainsi collectés ont besoin d'être annotés et compte tenu de la quantité de données à considérer certains ont recours à une externalisation à grande échelle pour l'annotation des objets et des positions des articulations du corps humain en utilisant la plateforme Amazon Mechanical Turk [Lin 2014a, Andriluka 2014].

Les performances des réseaux d'apprentissage profond pour l'apprentissage de caractéristiques pertinentes sont dépendantes de l'existence de grands jeux de données. De tels jeux de données ont permis à la communauté Vision de développer des réseaux profonds puissants et

adaptés aux problématiques de reconnaissance de postures humaines et d'objets.

Nous proposons de tirer parti de l'existence des détecteurs de la littérature pour pré-traiter les vidéos avant d'effectuer la reconnaissance d'actions. En effet cela nous permet d'extraire les observations de la posture humaine et des objets de la scène. Ces percepts de haut-niveau liés au contexte nous semblent pertinents.

1.2 Challenges

Nous présentons des challenges associés à la problématique de la reconnaissance d'actions par vidéo.

Challenges liés aux classes d'actions

Une grande variabilité intra-classe est due à de nombreux facteurs et elle peut être une conséquence des conditions d'enregistrement. Par exemple une action n'est pas perçue de la même manière en fonction du point de vue de la caméra/scène. Les conditions d'éclairage ou la présence de mouvement de caméra sont également des facteurs. Mais elle est aussi due aux légères différences qui existent entre deux individus dans l'exécution d'une même action. En effet ne serait-ce que de part la préférence manuelle qui varie selon les individus qui peuvent être droitiers, gauchers ou ambidextres. D'autres facteurs : l'encombrement de l'arrière-plan, variabilité de la taille des objets en fonction des effets de perspective.

La variabilité inter-classes exprime la difficulté de différencier certaines classes qui présentent des similarités entre elles. C'est un problème qui se pose dans la reconnaissance d'actions fines telles que l'action "boire une tasse" vs "boire une canette". À partir de l'observation seule du squelette, certaines actions peuvent être confondues comme "manger une pomme" ou "boire un verre d'eau".

Il est possible lors de l'exécution d'une action que certains membres du corps se retrouvent en situation d'auto-occultations. Aussi, dans le cas de manipulation de petits objets, ces objets peuvent être occultés par la main.

Challenges liés à la reconnaissance

La reconnaissance d'actions sur vidéos peut s'effectuer de différentes manières : sur des segments vidéos isolés qui ne contiennent que l'action à reconnaître ou sur des séquences vidéos au sein desquelles différentes actions se succèdent. De nombreux jeux de données proposent des vidéos pré-segmentées. Un des verrous scientifiques est d'effectuer la reconnaissance d'actions sans pré-segmentation (*untrimmed*).

À cause de la variabilité inter et intra-classes évoquée précédemment et de la taille de leur modèle, les réseaux d'apprentissage profonds ont besoin de nombreuses données annotées. Alors que les données vidéo peuvent exister sur le web, il reste à mener un lourd processus d'annotations. En effet les plus grands jeux de données ont souvent recours à la plateforme Amazon Mechanical Turk. De plus pour certaines applications ou encore pour des capteurs spécifiques,

une telle quantité de vidéos n'est pas toujours disponible. Et lorsque de tels jeux de données sont enregistrés souvent la répartition des classes n'y est pas toujours équitable et certaines classes sont sous-représentées.

Une autre remarque qui n'est pas spécifique à la reconnaissance d'actions mais plus généralement aux algorithmes de vision par ordinateur. Souvent ils requièrent le réglage d'hyper-paramètres. Ces paramètres sont externes au processus d'apprentissage mais il faut néanmoins les définir. La manière de choisir les hyper-paramètres a un impact sur les performances d'une approche donnée.

Dans une approche pragmatique, afin d'obtenir des observations de la posture et des objets, nous privilégions l'utilisation de réseaux d'apprentissage profond de la littérature. Ces réseaux sont nombreux et offrent d'excellentes performances. De plus nous allons explorer l'utilisation d'image couleur et/ou de profondeur. Divers indices contextuels sont pris en compte : les objets, l'affordance, leurs interactions et la temporalité. Parmi les approches mentionnées précédemment certaines ont besoin d'être embarquées donc nous proposons des approches modulaires qui restent compactes pour la partie reconnaissance. Nos travaux ne sont pas en lien avec un contexte applicatif particulier donc nous allons nous évaluer sur des jeux de données publics.

1.3 Feuille de route

Les différentes pistes d'investigation de cette thèse sont les suivantes.

Dans le cadre du projet DGA-RAPID SERVAT [1], nous menons une étude des outils de réglage (*tuning*) des hyper-paramètres appliqué à une application vision. Le but de cette étude comparative est de faire émerger quelques lignes directrices et des recommandations d'outils d'optimisation de la littérature à utiliser. En particulier lorsque nous sommes en présence d'un certain nombre de paramètres et d'un algorithme de nature stochastique à optimiser.

Nous proposons ensuite une approche de reconnaissance d'actions reposant sur la perception 3D conjointe de la posture humaine et des objets de la scène. Cette approche repose sur une modélisation bayésienne pour chacune de actions à reconnaître. Nous privilégions ce type d'approche car les vidéos dont il est possible d'obtenir les cartes de profondeur associées ne sont pas présentes sur les services de diffusion tels que YouTube.

Les mécanismes de fusion sont souvent utilisés pour améliorer les performances. Nous proposons de fusionner l'approche décrite précédemment avec un réseau de la littérature. Cette fusion nous permet d'étudier la complémentarité et la spécificité de chacune des approches et d'en tirer des bénéfices.

Pour finir et suite aux récentes avancées dans la définition des convolutions appliquées à des graphes. Nous proposons une approche compacte qui repose sur la définition de deux graphes : l'un représentant la posture humaine à travers un squelette et l'autre représentant les objets et leurs interactions possibles.

À noter que le début de la thèse a été financé par un projet DGA-RAPID sur la vidéo-surveillance (en lien avec le chapitre 3), puis elle est partiellement financée par un projet PIA3

1. Suivi Et Reconnaissance Visuels, Adaptatifs, Temps-réel

LinTo sur l'interaction en réunion de personnes et perception de leurs comportements.

1.4 Plan du mémoire

Le mémoire s'organise comme suit : le chapitre 2 présente une sélection d'approches variées de l'état de l'art pour la reconnaissance d'actions et permet de positionner nos travaux et justifier nos choix. Le chapitre 3 est consacré à l'étude comparative des outils de paramétrage des hyper-paramètres pour les algorithmes de Vision par ordinateur. Cette étude, demandée dans le cadre du projet DGA-RAPID, est destinée à donner quelques recommandations dans l'emploi de ces outils. Le chapitre 4 présente notre approche bayésienne pour la reconnaissance d'actions. Dans le chapitre 5 nous proposons de fusionner afin de tirer parti de deux types d'approches différentes ayant chacune leur spécificité. Le chapitre 6 propose de répondre à la problématique en utilisant deux réseaux à convolution sur les graphes. Enfin dans la conclusion nous y résumons et discutons des principales contributions de cette thèse puis nous proposons quelques perspectives prometteuses pour des travaux futurs.

CHAPITRE 2

État de l'art

Sommaire

2.1 Introduction

2.1.1 Enjeux et applications associées

La reconnaissance ou l'interprétation des actions humaines est requise pour le développement de nombreuses applications pour la vidéo-surveillance, l'interaction H/M, la cobotique mais également pour l'annotation et la recherche vidéo (*video retrieval*) ou encore dans le domaine de la réalité augmentée. Les activités réalisées à domicile ou dans un espace industriel peuvent différer mais leurs actions sous-jacentes peuvent être similaires car elles couvrent le déplacement d'objets, la saisie d'objets, l'interaction avec des objets...

La communauté Vision a porté une attention grandissante à ces applications avec l'émergence de challenges ou de jeux de données publics de plus en plus difficiles. En commençant d'abord par des défis de reconnaissance d'actions sur images ou de courts segments vidéos avec quelques classes. Puis en proposant des séquences vidéos plus longues avec un plus grand nombre de classes, des actions qui se succèdent ou encore des actions concomitantes.

Il s'agit toujours d'un verrou scientifique où les actions pourraient être mal interprétées en raison de l'encombrement du fond de la scène, des auto-occultations corporelles, de la présence de nombreux objets dans le voisinage humain et des changements de point de vue caméra/scène. De plus, une même action peut être exécutée différemment en fonction des personnes (variabilité intra-classe) et deux actions différentes peuvent se ressembler (variabilité inter-classe).

Certaines de ces difficultés peuvent être appréhendées par l'utilisation de capteurs et modalités supplémentaires, par exemple des dispositifs portables tels que des accéléromètres [Chen 2015]. La possibilité d'utiliser de tels équipements invasifs dépend du contexte ou n'est parfois pas envisageable. Ici nous nous focalisons sur les systèmes non portables et privilégions les capteurs optiques. Plus précisément les configurations où le capteur optique est

installé dans la pièce, en opposition aux configurations où le capteur optique est porté par les participants.

2.1.2 Définitions

Nous définissions dans cette section certains termes employés dans cet état de l'art et au travers des différents chapitres pour adresser la problématique de la reconnaissance d'actions.

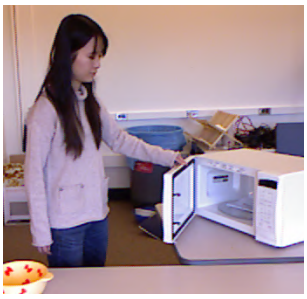
Geste, action, activité - Par le terme d'action nous nous référons, à l'instar de P. Turaga *et al.* [Turaga 2008], à des schémas de mouvements simples généralement exécutés par une seule personne et qui durent généralement de courtes périodes, de l'ordre de quelques dizaines de secondes. Par exemple : marcher, boire, téléphoner, etc. À la différence des gestes [Aggarwal 2011] qui eux sont définis comme des mouvements de base de parties du corps individuelles, par exemple : faire un signe de la main, hocher la tête etc. Les gestes ne font pas intervenir les objets et leur durée d'exécution est généralement plus courte. Les activités sont situées à un plus haut niveau, elles durent plus longtemps et mettent en jeux diverses actions qui se succèdent dans le temps. Par exemple l'activité *préparer le café* peut être considérée comme la séquence temporelle d'actions : *verser* l'eau, *ajouter* le café moulu et *allumer* la machine.

Affordance - Le concept d'affordance tel qu'introduit par J.J. Gibson [Gibson 1977] caractérise les propriétés fonctionnelles des objets et ce concept est largement repris dans la communauté robotique. En tant que telles, les affordances constituent un outil puissant d'apprentissage, de raisonnement et de prise de décision comportementale pour les systèmes artificiels. Par exemple les affordances que l'on peut attribuer à l'objet verre sont : déplacable, lavable, buvable. Cette notion d'affordance permet de lever certaines ambiguïtés lorsqu'elles se présentent. Pour reprendre l'exemple précédent il n'est pas possible de découper un verre.

Capteur - Il permet la collecte des données d'un jeu de données particulier. Un jeu de données multi-capteurs comporte plus de deux capteurs, dispositifs permettant d'enregistrer un jeu de données mais en augmente la complexité. Des exemples de capteurs sont : une caméra vidéo, une caméra à capteur de profondeur (RGB-D), Motion Capture (MoCap) avec suivi de marqueurs, accéléromètre, microphone.

Modalité, type de données - Des exemples de modalités sont : couleur, la profondeur, le flot optique, l'audio. Un système peut exploiter diverses modalité pour plus de robustesse. À la différence des données RGB qui encodent des informations liées à l'apparence, les données de profondeur (D pour *Depth*) sont géométriques c-à-d une mesure de la distance, en millimètres, des objets et des personnes dans le champ de vision du capteur RGB-D. Les images de profondeur peuvent être utilisées pour pour représenter géométriquement la scène observée et faciliter la segmentation de l'arrière-plan.

(a) Image - faire du vélo



(b) Trame vidéo - ouvrir

(c) Trame vidéo - fermer

FIGURE 2.1 – Exemples d'images issues de jeux de données image (Pascal VOC 2012) ou vidéo (CAD-120) pour la reconnaissance d'actions dans les images où dans les vidéos.

Séquence, clip - Les vidéos des différents jeux de données peuvent être appelées séquences vidéo par opposition aux clips qui en constituent une sous-partie.

2.1.3 Discussion

Les définitions précédentes nous permettent de préciser les contours de notre problématique. Dans cette thèse nous choisissons d'aborder la reconnaissance d'actions par l'utilisation de capteur visuels car ils sont largement usités et contiennent des informations riches et discriminantes. Dans nos approches nous allons exploiter les informations d'apparence et géométriques de la scène, nous privilégions donc les capteurs RGB-D car ils délivrent ces deux types de flux. En effet l'utilisation des informations de profondeur permet de percevoir la scène en 3D et ainsi d'être plus robuste au point de vue de la caméra par rapport à la scène. L'utilisation de cartes de profondeur permet également de segmenter l'arrière plan plus facilement en utilisant les données de distance. Nous choisissons d'étudier les actions de la vie courante qui sont exécutées par un seul individu, dans un environnement intérieur avec un capteur fixe. L'environnement est riche et présente divers objets avec lesquels interagir. Il y est donc possible d'avoir de riches actions d'interactions homme-objet et objet-objet. Les objets mis en jeu peuvent présenter des affordances variées qui aident a priori à discriminer les actions.

2.2 Reconnaissance d'actions basée image vs vidéo

La reconnaissance d'actions peut s'effectuer via la localisation sur une image de certains objets à l'instar de Zhou *et al.* [Zhou 2016] ou Oquab *et al.* [Oquab 2014]. Cette approche a été popularisée par le défi Pascal VOC 2012 qui propose de reconnaître 10 actions sur des images [Everingham 2010]. Le défi se poursuit avec la création d'un autre jeu de données MPII Human Pose 2014 [Andriluka] qui contient 400 actions à reconnaître sur un ensemble de 24000 images. Puisqu'il n'y a que l'information spatiale qui est exploitable sur une image, certains travaux [Yao 2012, Sharma 2013] proposent d'utiliser des percepts de haut niveau plutôt que des

caractéristiques de bas niveau. Parmi ces percepts de haut niveau nous pouvons retrouver : la posture humaine qui peut-être représentée plus ou moins finement, les objets détectés, les interactions homme-objets et des informations liées à la scène.

Certaines actions ne peuvent pas être discriminées seulement par l'utilisation d'image seule et il est nécessaire de la percevoir en vidéo. En présence de données vidéo il est ainsi possible d'exploiter la spatio-temporalité. Par exemple des caractéristiques issues des trajectoires suivies par les articulations associées à la posture humaine, typiquement les mains, au cours d'une vidéo peuvent être utilisées pour représenter les actions. Sur la figure 2.1 nous pouvons voir sur l'image 2.1a que la personne fait du vélo alors que les actions ouvrir et fermer des images des figures 2.1b et 2.1c sont difficiles à discriminer sans la connaissance des mouvements qui ont eu lieu sur avant. Une approche de reconnaissance d'actions représentative basée sur ces trajectoires est la méthode *improved Dense Trajectories (iDT)* proposée par [Wang 2013a]. L'évolution temporelle de caractéristiques liées à l'apparence ou à la posture peut être modélisée par des automates à états tels que le modèle de Markov caché (MMC) [Ikizler 2007] ou un champ aléatoire conditionnel [Hu 2014, Koppula 2013b]. Plus récemment, les approches d'apprentissage profond (*deep-learning*) ont fait l'objet d'une attention grandissante de part leur succès pour l'apprentissage de caractéristiques avec de bonnes capacités de généralisation et l'émergence de grands jeux de données vidéos. Les trois composantes majeures de leur développement pour la reconnaissance d'action vidéo sont l'introduction d'opérations de convolutions avec un axe temporel [Tran 2015, Carreira 2017], les réseaux récurrents pour modéliser les relations temporelles à long terme [Jain 2016a] et l'introduction des réseaux de convolutions sur des graphes [Kipf 2017].

Si la reconnaissance d'actions sur image est pertinente lorsque les classes d'actions à reconnaître interviennent dans des environnements différents, ces approches sont inappropriées pour reconnaître les actions atomiques qui se succèdent dans la même scène. Ce sont les mouvements mis en jeux et les affordances lors de l'exécution d'une action qui permettent de la discriminer, par exemple lors de l'ouverture ou de la fermeture d'une porte. C'est pourquoi nous nous focalisons sur les approches utilisant des informations spatio-temporelles à partir de vidéos afin de considérer la dynamique du corps mais aussi des objets lors des actions. Nous proposons de classer ces approches dites dynamiques suivant si elles appartiennent à la famille des dynamiques en approches paramétriques ou approches d'apprentissage profond comme nous pouvons le voir sur la taxonomie décrite par la figure 2.2. Dans un deuxième temps nous différencierons différentes approches suivant le type d'informations qu'elles fusionnent : caractéristiques multi-capteurs, caractéristiques mono-capteurs ou informations symboliques. Un dénominateur commun à ces approches est la perception conjointe de l'homme et des objets. L'état de l'art présenté ci-après n'est pas exhaustif et pour plus de détails nous pouvons mentionner ces articles [Herath 2017, Yao 2019].

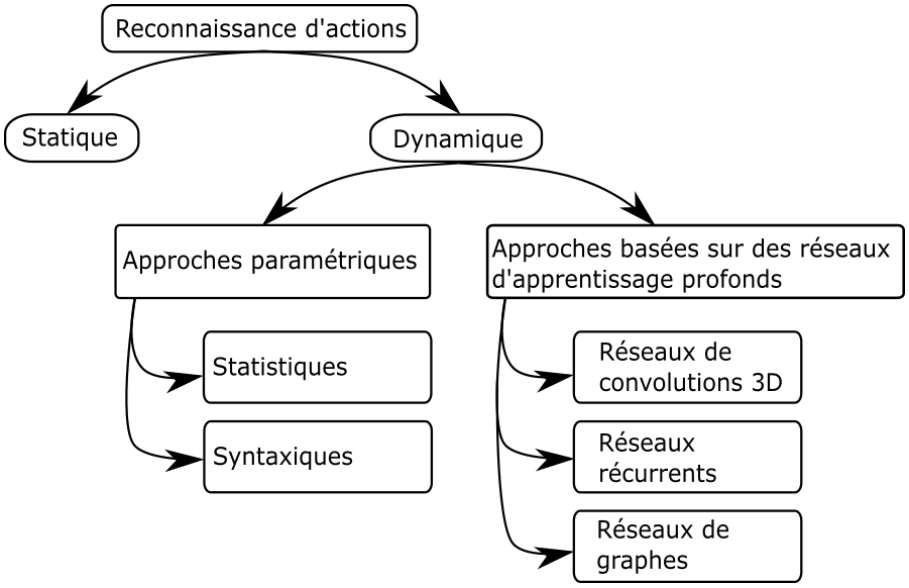


FIGURE 2.2 – Taxonomie suivie lors de ce chapitre.

2.3 Approches paramétriques vs approches d'apprentissage profond

Nous décrivons quelques approches paramétriques de l'état de l'art dans la section 2.3.1, puis des approches d'apprentissage profond dans la section 2.3.2 et enfin une discussion en section 2.3.3.

2.3.1 Approches paramétriques

Statistiques - Parmi les approches statistiques nous pouvons mentionner les travaux de H. Koppula et A. Saxena [Koppula 2013a]. Ils emploient un champ aléatoire de Markov pour modéliser les interactions, où les nœuds du graphe représentent les objets et les mouvements du corps humain et les arcs représentent une relation. Leur approche infère les actions à partir des squelettes extraits du capteur RGB-D, de la nature et la position des objets pour reconnaître les actions et les activités de la vie courante. N. Hu *et al.* [Hu 2014] proposent d'utilisation d'un champ aléatoire conditionnel latent pour encoder les mêmes relations que [Koppula 2013a]. À la différence que leur modèle est plus flexible et encode l'ensemble des connections entre les observations et les noeuds d'état dans un graphe complet. Ils évitent ainsi de faire des hypothèses d'indépendance conditionnelles qui seraient inappropriées. Ces deux approches caractérisent les actions de manière plus explicite à travers une modélisation des observations des éléments de la scène.

Syntaxiques - Des modèles de grammaire stochastique ont été utilisés pour analyser les structures hiérarchiques des vidéos impliquant des humains. Qi *et al.* [Qi 2017] représente les

affordances, les actions humaines et les objets en interaction dans un graphe spatio-temporel "And-Or" (ST-AOG) pour prédire les activités humaines dans les vidéos RGB-D. Ils construisent un modèle en deux étapes principales : (i) l'analyse de la vidéo, et (ii) la prédiction de l'activité. L'analyse syntaxique est effectuée par segmentation au moyen d'une approche de programmation dynamique, puis par affinement au moyen d'un échantillonnage de Gibbs. Pour la prédiction de l'activité, l'approche repose sur un analyseur (*parser*) Earley [Earley 1970] pour prédire les actions. Les méthodes basées sur les grammaires sont appropriées pour répondre aux problématiques qui présentent des structures composées. Cependant, [Qi 2017] prennent en entrée des éléments symboliques, qu'il faut au préalable détecter, comme les analyseurs de langage traditionnels.

Les trois approches paramétriques décrites ci-dessus [Koppula 2013a, Hu 2014, Qi 2017], sont considérées comme des approches benchmark dans les chapitres suivants.

2.3.2 Approches reposants sur l'apprentissage profond

Réseaux avec convolutions 3D - Une manière d'adresser le problème de la reconnaissance d'actions avec des réseaux d'apprentissage profond est d'ajouter des informations temporelles aux informations spatiales lors de l'opération de convolution. Un réseau de neurones à convolution 3D utilise des filtres 3D (filtres étendus le long de l'axe du temps) pour extraire des caractéristiques des dimensions spatiales et temporelles. Il est donc censé capturer des informations spatio-temporelles et des mouvements encodés dans des images successives. De manière générale, les réseaux de neurones à convolution 3D ont une structure temporelle très rigide. En effet, le réseau accepte un nombre prédéfini de trames en entrée. Une illustration d'un tel réseau est présentée par (b) sur la figure 2.4. C3D [Tran 2015] est un réseau avec de telles convolutions 3D pour la reconnaissance d'actions. La taille du modèle C3D est d'environ 15 millions de paramètres.

Il est difficile d'entraîner des réseaux d'apprentissage profond sur des vidéos car il y a besoin de nombreuses données annotées, et il serait donc préférable de bénéficier de modèles pré-entraînés sur de larges jeu de données images (par exemple ImageNet [Deng 2009]). En outre, les modèles entraînés sur les images ont de bonnes performances pour la compréhension des relations spatiales des objets et cette capacité pourrait également être exploitée pour la reconnaissance d'actions. C'est cette idée de transfert de connaissances de modèles d'images vers un modèle vidéo qui est examinée par I3D [Carreira 2017] au travers de la technique de l'*inflation* qu'ils décrivent.

Réseaux récurrents - A. Jain *et al.* [Jain 2016a] proposent un réseau de neurones récurrent structuré (S-RNN) avec des structures de graphes spatio-temporels prédéfinies. Ils proposent une mise en correspondance (*mapping*) d'un graphe spatio-temporel à un mélange d'architecture de réseau de neurones récurrents où les nœuds et les arêtes sont représentés par des cellules de mémoire à long terme et à court terme (LSTM). Ils s'appuient essentiellement sur deux RNN indépendants : edgeRNN pour modéliser l'évolution des interactions dans le temps et nodeRNN

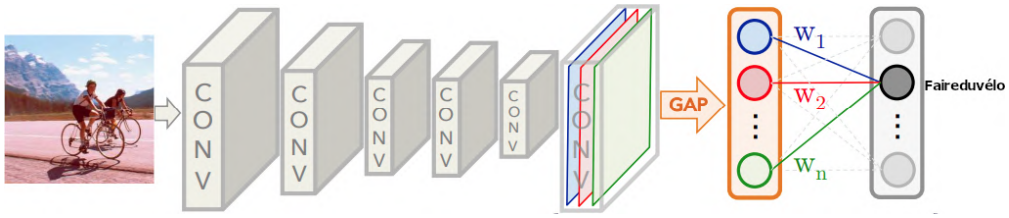


FIGURE 2.3 – Apprentissage de bout-en-bout à partir d'une image vers la prédiction d'action. Conv désigne des couches de convolutions et w des poids appris par le réseau. Illustration du réseau par Oquab *et al.* [Oquab 2014].

pour l'inférence spatiale. Ils raisonnent sur des squelettes 3D et leur interaction avec des objets. Le graphe spatio-temporel prédéfini change en fonction de la tâche. Par exemple, boire n'implique que le nœud humain et le nœud de la tasse, alors que l'action de préparation des céréales implique un nœud pour l'homme et des nœuds pour le bol, le lait et les céréales. Cependant, ils relient tous les objets entre eux dans leur graphe spatio-temporel afin de gérer le nombre dynamique d'objets dans les différentes séquences. Ils gèrent la variété des graphes spatio-temporels dans un seul graphe structurel RNN.

Réseaux d'apprentissage profond et graphes - Pour apporter des propriétés plus génériques, S. Qi *et al.* [Qi 2018b] proposent d'apprendre la structure du graphe au lieu d'utiliser des structures prédéfinies. Le réseau apprend à déduire la matrice d'adjacence (qui représente le graphe) de bout en bout. Ils s'appuient sur un réseau de neurones de graphes (GNN). Grâce à de multiples itérations de passage de messages et de mise à jour des états des nœuds, chaque nœud capture la relation sémantique et les informations structurelles au sein de ses nœuds voisins. Ils utilisent une boîte englobante pour représenter l'homme et les objets de la scène.

Le modèle sous-jacent de [Qi 2018b] (GPNN) est un *Graph Neural Network* (GNN). Des études récentes ont proposé une variante aux GNNs : le réseau convolutionnel de graphes (GCN). Les convolutions sont adaptées d'une grille 2D ou 3D à une structure de graphe. Cette idée a été initialement appliquée à la reconnaissance d'actions avec le ST-GCN [Yan 2018]. Les nœuds du ST-GCN correspondent aux articulations du corps humain, qui sont connectées dans l'espace et dans le temps. Les bords spatiaux relient les articulations en suivant la structure du squelette humain, et les bords temporels relient le même type de nœud à travers le temps. Ils utilisent uniquement les informations du squelette pour déduire des actions. Le ST-GCN est plus efficace en termes de calcul que les réseaux neuronaux convolutifs plus classiques, notamment en ce qui concerne l'utilisation de la mémoire, grâce à l'introduction des convolutions de graphes.

Approches dites end-to-end vs par pré-traitements du flux - Les approches d'aprentissage profond de bout-en-bout (*end-to-end*) permettent de s'affranchir de nombreuses étapes de prétraitement. Au prix d'une taille de réseau plus grande, d'une dépendance à la disponibilité d'un grand jeu de données et d'une difficulté accrue dans l'interprétation des caractéristiques apprises. Par exemple sur la figure 2.3 on peut voir que l'image du jeu de donnée PASCAL VOC 2012 est

directement donnée en entrée du réseau afin de prédire l'action faire du vélo. Parmi les approches mentionnées, les réseaux C3D et I3D sont ce type de réseau end-to-end.

Une autre alternative est d'utiliser quelques pré-traitements du flux vidéo afin de décomposer le problème de reconnaissance d'actions en sous-problèmes. Pour reprendre l'exemple de la figure 2.3, il est également possible d'abord détecter la personne ou sa posture et les objets et ensuite d'apprendre un modèle pour discriminer les différentes classes d'actions. Le premier avantage c'est qu'il existe de nombreuses données pour la reconnaissance d'objets [Liu 2016a, Girshick 2015, He 2017, Lin 2017, Redmon 2018] et de posture [Toshev 2014, Insafutdinov 2016, Cao 2017, Alp Güler 2018] et des modèles de réseaux pré-entraînés sont disponibles. Le deuxième c'est la possibilité d'ajouter d'autres caractéristiques ou relations entre les données pré-traitées. Un autre enjeu est ainsi de proposer des réseaux plus compactes avec des temps d'apprentissage plus faibles dont on peut bien régler ses hyper-paramètres. Les approches [Koppula 2013a, Hu 2014, Qi 2017, Jain 2016a, Qi 2018b, Yan 2018] requièrent un prétraitement de flux vidéo.

2.3.3 Discussion

Les réseaux d'apprentissage profond sont proéminents dans la littérature sur la reconnaissance des actions, mais les méthodes paramétriques restent pertinentes. Par rapport aux réseaux profonds, elles présentent l'avantage d'être plus faciles à entraîner car elles ont moins de paramètres et donnent généralement de bonnes performances sur de petits jeux de données. A contrario, les approches d'apprentissage profond de bout-en-bout restent dépendantes de l'existence de grands jeux de données. C'est aussi l'une des raisons pour lesquelles les approches qui reposent d'abord sur des pré-traitements sont nombreuses et surtout lorsque des données RGB-D sont utilisées. En effet les bases de données RGB-D sont généralement plus petites car ce type de données ne peut pas être collectées à partir de données de films ou depuis Youtube par exemple.

À notre connaissance, peu de travaux proposent de fusionner ces deux types d'approches (paramétriques et d'apprentissage profond) alors qu'elles pourraient être complémentaires sur certains jeux de données qui présentent des difficultés liées à leur taille et/ou la répartition statistique de leurs classes. Les récentes approches des réseaux avec graphes [Qi 2018b, Yan 2018] semblent prometteuses et adaptées à notre problématique et il serait intéressant d'explorer comment on peut y ajouter d'autres caractéristiques ou de créer de nouveaux réseaux ▷ comment insérer la perception des objets.

2.4 Approches avec stratégies de fusion

Nous décrivons ci-après des approches qui pour aborder la reconnaissance d'actions envisagent l'emploi et la fusion d'informations émergeant de multiples capteurs dans la section 2.4.1, des approches qui emploient la fusion de multiple modalités d'un même capteur dans section 2.4.2, puis la fusion de données symboliques dans la section 2.4.3 et enfin une discussion en section 2.4.4.

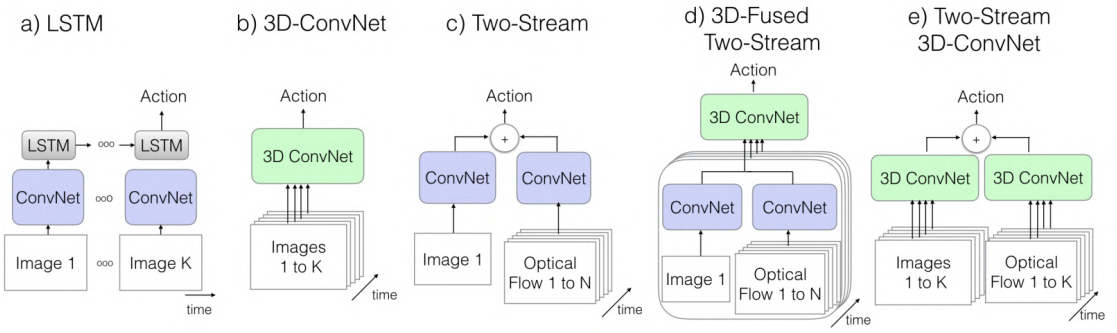


FIGURE 2.4 – Présentation de divers réseaux de la littérature avec/sans fusion, illustration issue de Carreira et Zisserman 2017 [Carreira 2017].

2.4.1 Fusion de percepts multi-capteurs

Afin d'apporter plus de robustesse, il est possible de multiplier les capteurs et les modalités. Les différentes caractéristiques telles que l'audio et la vidéo peuvent être fusionnées de manière tardive à l'instar de [Kazakos 2019] pour de la reconnaissance d'actions égocentrique (c-à-d où le capteur vidéo filme les champs de vision de la personne qui exécute l'action). Ou encore dans [Fan 2016] pour la reconnaissance d'émotions, ils proposent une fusion tardive des caractéristiques du réseau C3D avec des caractéristiques audio. D'autres types de capteurs sont envisagés, souvent des accessoires de type technologie portable (*wearable technology*) tels que les accéléromètres, gyroscopes, capteurs de pression (pour les mouvements du corps et les forces appliquées... Les données de multiples capteurs sont pré-traités pour synchronisation et normalisation des différents capteurs. Par exemple [Chen 2015] proposent d'utiliser une caméra de profondeur et des accéléromètres pour la reconnaissance d'actions.

2.4.2 Fusion de percepts implicites mono-capteur

Il est parfois possible à partir d'une seul capteur d'extraire différentes modalités, c'est le cas pour une caméra RGB dont nous pouvons extraire les images couleurs mais également calculer le flot optique. Certaines approches telle que [Feichtenhofer 2016] proposent une fusion spatio-temporelle à partir d'une seule image couleur qui est fusionnée avec sa séquence de flot optique associée. L'image décrit les caractéristique spatiales et la séquence de flot optique qui décrit le mouvement est là pour la dimension temporelle. Cette approche est illustrée par (c) sur la figure 2.4. Cette idée est reprise par I3D, tel qu'illustré par (e) sur la figure 2.4, où l'on peut voir qu'ici ce n'est pas qu'une seule image qui est fusionnée avec la séquence de flot optique, mais bien une séquence d'images. Cela améliore les performances par rapport à un réseau C3D seul qui chercher à extraire simultanément les caractéristiques temporelles et spatiales au niveau des couches de convolution 3D. Ce gain de performance se fait au détriment de la taille du réseau, en effet I3D a environ 25 millions de paramètres à comparer aux 15 millions de C3D.

Pour gérer la taille limitée des jeux de données, H. Kataoka *et al.* [Kataoka 2016] utilisent

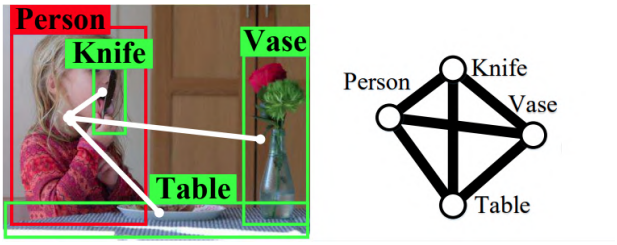


FIGURE 2.5 – Percepts explicites fusionnés ici : personne, couteau, vase et table via leurs segmentations préalables (boîtes englobantes). Modélisation par un graphe. Illustration issue de S. Qi *et al.* [Qi 2018b].

des images couleurs et différentielles comme entrée d'un CNN pré-entraîné sur ImageNet (plus d'un million d'annotations) pour effectuer de la détection d'actions sur Watch-n-Patch. Ils s'évaluent sur des jeux données qui contiennent les informations de profondeur mais il ne se basent que sur les images RGB.

M. S. Ryoo et al. [Ryoo 2015] combinent des caractéristiques CNN 2D pré-entraînées sur ImageNet et des caractéristiques de mouvement conventionnelles telles que les histogrammes de flux optique (HOF). Cette approche ne repose que sur des images et des caractéristiques de mouvement en 2D. Ils effectuent une fusion précoce de caractéristiques liées au flot optique et des caractéristiques apprises liées à l'apparence. Ils soulignent l'importance des motifs temporels et d'une représentation à long terme des dynamiques pour la reconnaissance d'actions. Ils s'évaluent sur des jeux données qui contiennent les informations de profondeur mais il n'emploient que les images RGB. Ces deux approches [Kataoka 2016, Ryoo 2015] sont considérées comme des benchmarks dans des chapitres suivants.

2.4.3 Fusion de percepts explicites mono-capteur

Comme précédemment, nous présentons des approches qui proposent une fusion à partir de données issues d'un seul capteur. Précédemment, nous avons vu la fusion de percepts plutôt bas-niveau tels que les images couleurs aux flots optiques ou à des descripteurs du flot optique comme les histogrammes HOF. À contrario, cette section se focalise sur une fusion percepts de plus haut niveau tels que des trajectoire de squelettes 2D ou 3D pour décrire l'évolution de la posture humaine, ou la position et la nature des objets de la scène. H. Koppula et A. Saxena dans [Koppula 2013b] via leur champ de Markov conditionnel considèrent les trajectoires de squelettes obtenus en utilisant une fonctionnalité proposée avec une bibliothèque associée au capteur de profondeur utilisé (OpenNI pour Kinect), ainsi que la position des objets. S. Qi *et al.* dans [Qi 2018b] proposent de raisonner sur la connaissance de la position de la personne et des objets représentés par une boîte englobante comme illustré sur la figure 2.5.

2.4.4 Discussion

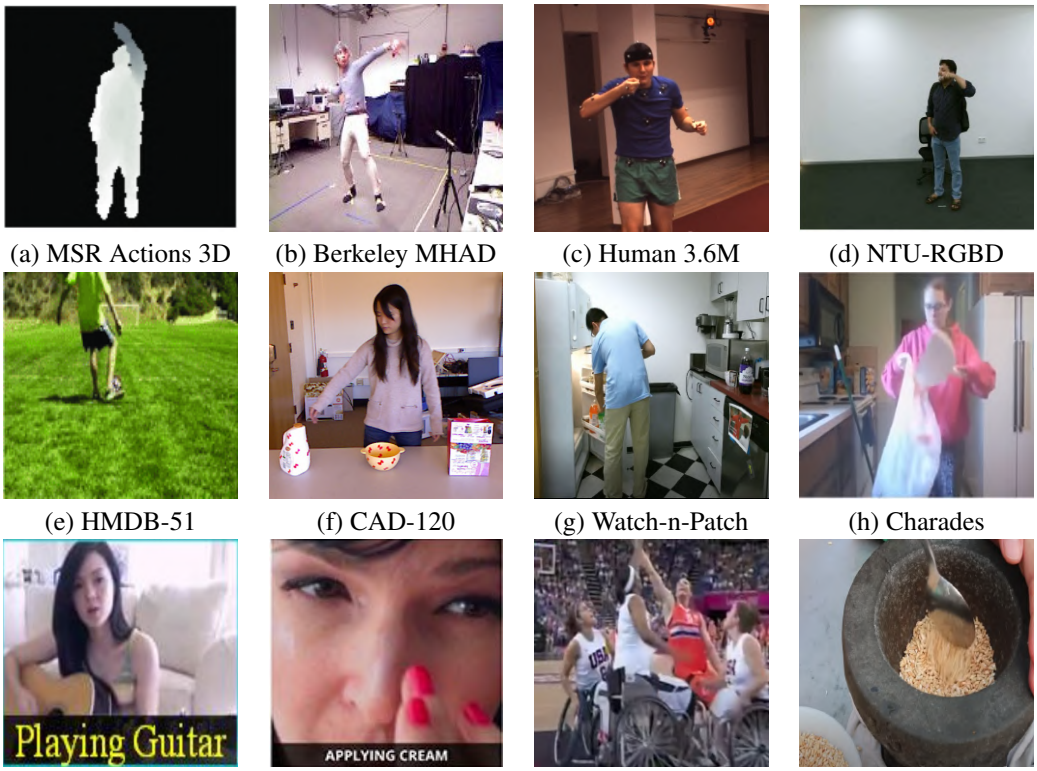
Des stratégies de fusion sont souvent proposées pour relever de nombreux défis. Par exemple, la fusion peut consister à diversifier les capteurs en utilisant des capteurs audio-vidéo ou encore à ajouter le flux optique qui décrit le mouvement général à une séquence d'images comme dans [Feichtenhofer 2016]. Ces différentes approches démontrent les avantages de l'utilisation d'un mécanisme de fusion de modalités ou de percepts bas-niveau pour augmenter la performance globale. Toutefois, ce gain est obtenu au détriment de la quantité de données nécessaires à l'entraînement. L'ajout d'un plus grand nombre de modalités augmente le nombre de paramètres à apprendre pour le réseau de convolution. Cela a deux effets : premièrement, cela nécessite l'existence d'un tel ensemble de données et deuxièmement, cela augmente le temps d'entraînement. En revanche, il est possible de proposer des approches qui demeurent compactes en fusionnant des informations de plus haut-niveau et ce grâce aux nombreux pré-traitements vidéos envisageables : détection de la posture humaine, des objets, voire des éléments contextuels autres.

TABLE 2.1 – Caractéristiques de certains jeux de données publics pour la reconnaissance d'actions.

	Date	Couleur (RGB)	Profondeur (D)	Nb de vidéos	Objets	Localisation	Nombre de classes	Environnement
MSR Action [Li 2010]	2010	×	✓	567	×	×	20	ctrl.
HMDB-51 [Kuehne 2011]	2011	✓	×	6849	✓	×	51	non ctrl.
UCF-101 [Soomro 2012a]	2012	✓	×	13000	✓	×	101	non ctrl.
Berkeley MHAD [Ofli 2013]	2013	✓	✓	660	×	×	11	ctrl.
CAD-120 [Koppula 2013b]	2013	✓	✓	120	✓	✓	10	ctrl.
Human 3.6M [Ionescu 2014]	2014	✓	✓	1320	×	×	15	ctrl.
Sports-1M [Karpathy 2014]	2014	✓	×	1133158	✓	×	487	non ctrl.
Watch-n-Patch [Wu 2015a]	2015	✓	✓	458	✓	✓	21	ctrl.
NTU-RGB+D [Shahroudy 2016]	2016	✓	✓	56880	✓	×	60	ctrl.
Charades [Sigurdsson 2016]	2016	✓	×	9848	✓	✓	30	non ctrl.
Youtube 8M [Abu-El-Haija 2016]	2016	✓	×	8M	✓	×	4716	non ctrl.
NTU RGB+D 120 [Liu 2019]	2019	✓	✓	114480	✓	×	120	ctrl.
Kinetics-700 [Carreira 2019]	2019	✓	×	650k	✓	×	700	non ctrl.

2.5 Jeux de données : motivations et choix associés

Cette section présente un panel de jeux de données publics pour la reconnaissance d'actions. Ce recensement est également motivé par le fait que nos travaux ne s'inscrivent pas dans un contexte applicatif particulier en dehors de notre problématique. Les approches de la littérature



(a) MSR Actions 3D (b) Berkeley MHAD (c) Human 3.6M (d) NTU-RGBD

(e) HMDB-51 (f) CAD-120 (g) Watch-n-Patch (h) Charades

(i) UCF-101 (j) Kinetics-700 (k) Sports 1M (l) YouTube 8M

FIGURE 2.6 – Illustration de quelques jeux de données.

sont souvent évaluées et comparées sur les jeux de données présentés et nous permettent ainsi de comparer nos travaux à la littérature. De plus l'acquisition et l'annotation d'un jeu de données est chronophage. En privilégiant des jeux publics, nous nous exonérons donc de cet exercice souvent fastidieux. Nous présentons dans le tableau 2.1 une synthèse des différentes caractéristiques des jeux. Les jeux de données sont décrits dans l'ordre chronologique de leur publication.

MSR Action 3D - Microsoft Research (MSR) Action 3D a été créé par [Li 2010]. Les vidéos ont été capturées à l'aide de capteurs de profondeur et sont composées de 10 personnes qui exécutent 20 actions telles que : coup de pied avant, lancer, ramasser, frapper, courir, service au tennis, etc. Comme nous pouvons le voir sur la figure 2.6a, seule la silhouette de la personne apparaît.

HMDB-51 - Les vidéos de ce jeu de données sont collectées avec une caméra couleur à partir de sources diverses telles que des films ou des vidéos amateur. Les 6849 vidéos appartiennent à 51 classes. Ces classes peuvent être regroupées en 5 types : actions liées aux mouvement faciaux (rire, parler), les mouvements faciaux avec des objets (fumer, manger), les gestes tels que applaudir, des actions avec interaction avec des objets et les actions avec interaction entre

personnes. Ce jeu de données est créé dans le but de développer des approches robustes afin de d'effectuer de l'indexation vidéo et les différentes actions ne sont pas localisées dans le temps. Une illustration est présentée sur la figure 2.6e

UCF-101 - Ce jeu de données de reconnaissance d'actions a été crée par [Soomro 2012a]. Les vidéos RGB sont collectées à partir de YouTube et donc l'environnement est non contrôlé. Ce jeu rassemble 101 classes d'actions, regroupées en 5 catégories : interaction homme objet, mouvement du corps, interactions entre personnes, jouer un instrument et les actions liées au sport. Des objets sont donc présents dans ce jeu de données avec par exemple : taper à l'ordinateur ou jouer de la guitare comme illustré par la figure 2.6i. L'inconvénient c'est que dans ce jeu de données la plupart des objets présents ne sont pas associés à de multiples actions. Ainsi l'objet guitare n'apparaît que dans les vidéos associées à l'action "jouer de la guitare". Une illustration est présentée sur la figure 2.6i

Berkeley MHAD - C'est un jeu de données multi-capteurs, enregistré dans une salle de laboratoire équipée d'un système de MoCap, 4 caméras vidéos, 2 capteurs de profondeur, 4 micros et 6 accéléromètres. 12 acteurs y jouent 11 actions faisant intervenir les mouvements du corps humain. Par exemple : s'asseoir, se lever, sauter, applaudir... Chaque acteur répète les actions 5 fois. Les actions sont jouées dans un environnement épuré, hormis la présence des capteurs et de l'acteur il n'y a pas d'autres informations ni objets comme nous pouvons le voir sur la figure 2.6b.

CAD-120 - Le jeu de données CAD-120 [Koppula 2013b] comprend 120 vidéos avec des canaux RGB-D, jouées par 4 acteurs. Il contient 10 activités de la vie quotidienne (préparation d'un bol de céréales, prise de médicaments...). Ces activités comportent 10 actions : atteindre, bouger, verser, manger, boire, placer, ouvrir, fermer, nul. Depuis sa sortie, il a été cité plus de 590 fois et plus de 100 fois au cours de l'année 2020, les approches [qi2] [Qi 2018b] s'évaluent également sur ce jeu de données. Il présente de riches interactions avec les objets, comme nous pouvons le voir sur la figure 2.6f avec la présence d'un bol, d'une bouteille et d'un four à micro-ondes sur une table.

Human 3.6M - Le jeu de données Human 3.6M [Ionescu 2014] consiste en 3.6 millions de postures humaines en 3D et leurs images correspondantes. Il est enregistré à partir d'une caméra couleur, d'un capteur de profondeur et d'un système de MoCap. 11 acteurs professionnels ont joué 17 actions différentes : prendre une photo, parler au téléphone, fumer, discuter... Cependant il ne contient aucun objet sauf parfois une chaise. Même pour l'action de manger comme illustré par la figure 2.6c, l'acteur mime l'action et n'a rien dans les mains.

Sports-1M - Il contient près d'un million de vidéos qui ont été automatiquement annotées en 487 classes. C'est l'un des plus grand jeu de données vidéos. Les vidéos n'ont pas été enregistrées spécifiquement pour ce jeu de données mais elles ont été collectées. L'environnement est

donc non contrôlé et elles contiennent donc de nombreux sports très variés, des mouvements de caméras ainsi qu'une grande variabilité intra-classe.

Watch-n-Patch - Ce jeu de données offre deux environnements avec des actions différentes et leurs ensembles d'entraînement et de tests associés pour un total de 21 actions. L'environnement de bureau se compose de 196 vidéos enregistrées dans 8 bureaux différents. Il y a 10 actions annotées : lire, marcher, quitter le bureau, aller chercher un livre, remettre un livre à sa place, poser un objet, ramasser un objet, jouer à l'ordinateur, allumer un ordinateur, éteindre un ordinateur. Une illustration de ce jeu de données est présentée sur la figure 2.6g. L'environnement de la cuisine se compose de 117 vidéos enregistrées dans 5 cuisines différentes. Il y a 11 actions annotées : aller chercher au réfrigérateur, remettre au réfrigérateur, préparer les aliments, faire cuire au micro-ondes, aller chercher au four, verser, boire, quitter la cuisine, remplir la bouilloire, brancher la bouilloire, déplacer la bouilloire. Dans ce jeu de données les actions sont agencées en séquences et donc localisées dans le temps et un objet donné peut apparaître dans différentes actions.

NTU-RGB+D - Ce jeu de données multi-modal est enregistré à partir 3 caméras couleurs et 3 capteurs de profondeur et d'un capteur infrarouge. Il propose 60 classes d'actions regroupées en 3 groupes : actions quotidiennes, actions reliées à la santé et des actions d'interactions entre personnes. 40 personnes d'âge, de genre et de taille différentes ont participé à l'enregistrement. Les actions ne sont pas localisées mais enregistrées séparément, indépendemment les unes des autres. Un exemple d'image pour l'action de téléphoner est présentée sur la figure 2.6d. Une nouvelle version de ce jeu de données NTU-RGB+D 120 [Liu 2019] a été publiée en 2019.

Charades - Charades [Sigurdsson 2016] est un jeu de données qui contient 9848 vidéos enregistrées par des volontaires via Amazon Mechanical Turk où ils devaient jouer un script. Au total 267 personnes différentes ont participé et se sont enregistrées. Le jeu de données présente 30 actions et 46 classes d'objets. Les scripts mentionnent des successions d'actions avec des objets à utiliser, de fait les participants enregistrent des vidéos dont des actions qui sont localisées dans le temps. L'image sur la figure 2.6h est issue du script suivant : une personne passe le balai et verse la poussière de la pelle dans un sac poubelle et le ferme.

YouTube-8M - Créé plutôt pour les applications de classification vidéos. Ce jeu contient 4716 actions dans de longues vidéos qui durent entre 120 et 500 secondes. Diverses actions au sein d'une même vidéo par exemple l'action choisie et illustrée par la figure 2.6l, représente l'action "cuisiner" alors qu'il y a de nombreuses sous-actions telles que : prendre du riz, prendre un mortier, verser le riz dans le mortier etc.

Kinetics-700 - Kinetics existe en 3 versions qui ont été publiées entre 2017 et 2019 : Kinetics-400, Kinectics-600 et Kinetics-700 [Carreira 2019]. Ces vidéos sont collectées à partir de sources variées sur YouTube. Chacune des 650000 vidéos dure approximativement 10 secondes

et ne contient qu'une seule action annotée. L'ensemble contient 700 classes d'actions telles que le fait de jouer un instrument, de se serrer la main, de jouer aux cartes ou de caresser un chat. Ce jeu de données présente des objets mais les actions réalisées ne sont pas agencées en séquence. La figure 2.6j montre une illustration.

Malgré la diversité des jeux de données existants dans la littérature, tous ne correspondent pas à notre problématique. Comme nous pouvons le voir sur le tableau récapitulatif 2.1, en effet certains ne contiennent pas d'objets et ne peuvent donc être considérés. Pour les jeux qui contiennent des objets comme UCF-101 et Kinetics-700, puisque les vidéos sont issues de sources variées il n'est pas possible de suivre l'évolution de la configuration des objets au travers de l'exécution de séquences d'actions. Pour pouvoir percevoir ces changements dans les objets et leur configuration il apparaît donc la nécessité d'utiliser des jeux de données dont les actions sont également localisées dans le temps. C'est pourquoi nous avons privilégié : CAD-120 et Watch-n-Patch puis Charades.

2.6 Conclusion

Nos travaux se focalisent sur la reconnaissance d'actions humaines avec des objets manipulables, dont les positions et interactions vont varier au cours de la vidéo. Nous exploitations la perception conjointe homme-objets au moyen d'un capteur vidéo et/ou de profondeur pour discriminer les actions. Les détections de la posture humaine et des objets sont obtenues en utilisant des détecteurs de la littérature.

Cette étude permet : (1) d'exhiber plusieurs voies d'investigations : paramétrique seul, apprentissage profond, approche de fusion et modélisation par des graphes, (2) d'identifier des approches de référence sur cette problématique à des fins de comparaison, (3) de décrire et d'identifier les jeux de données puis de motiver nos choix, il est important de disposer de jeux de données différents pour vérifier la répétabilité des performances, (4) les données RGB-D nous permettent par la suite d'exploiter indifféremment l'apparence et/ou la géométrie de la scène.

Nombreuses approches reposent sur un ou plusieurs paramètres à régler. Nous commençons donc, dans le chapitre suivant, par présenter notre étude des outils d'optimisation des hyperparamètres pour un algorithme de vision par ordinateur.

CHAPITRE 3

Vision et optimisation des hyper-paramètres

Sommaire

3.1 Introduction

L'objectif principal de ce chapitre est de fournir au lecteur une analyse, aussi détaillée que possible, de plusieurs outils d'optimisation des hyper-paramètres. Il met en évidence les points forts, les points faibles à considérer et les autres aspects d'intérêt qui peuvent motiver le lecteur à choisir l'outil le plus pertinent, pour des algorithmes de vision par ordinateur et plus principalement dans un contexte de suivi multi-objets. L'analyse se concentre sur 8 points importants qui rendent les outils efficaces et pratiques à utiliser : (1) Vitesse de convergence, (2) Stabilité, (3) Précision, (4) Temps de calcul, (5) Robustesse aux valeurs initiales, (6) Taille de l'entraînement et (7) Nombre de particules, et (8) Documentation.

Dans la section 3.2 nous présentons les différents outils analysés, puis nous exposons l'algorithme de suivi multi-objets retenu et ses paramètres associés dans la section 3.3 puis l'ensemble des évaluations est présenté dans la section 3.4 et pour finir une conclusion sur ces travaux.

3.2 Stratégies d'optimisation

Ce chapitre propose une analyse de différents outils de la littérature et montre comment nous pouvons les adapter à une problématique de la communauté Vision considérée comme une fonction boîte noire.

Il existe plusieurs stratégies d'optimisation des hyper-paramètres dans la littérature appartenant à différentes classes [Bergstra 2011, BoussaïD 2013, Hutter 2014]. Nous nous concentrons sur les approches stochastiques basées sur l'optimisation bayésienne. Cette méthodologie permet d'estimer le maximum (ou le minimum) global d'une fonction boîte noire bruitée en développant des modèles statistiques. Elle s'est avérée être une solution puissante dans divers domaines tels que l'apprentissage automatique [Luo 2016, Snoek 2012], la robotique, l'apprentissage profond [Domhan 2015], l'optimisation combinatoire [Wang 2013b, Hutter 2011b], parmi tant d'autres.

Notre objectif est d'étudier l'impact de ce type d'optimisation sur les performances d'un algorithme de vision par ordinateur dépendant d'hyper-paramètres. Nous avons besoin d'une telle fonction boîte-noire et d'une fonction-objectif qui permet d'en mesurer les performances. De nombreuses fonctions de vision peuvent être considérées comme une telle fonction boîte-noire, par exemple : un réseau de reconnaissance d'objets, un algorithme de segmentation, des approches de suivi multi-objet...

Les métriques de performance qui les évaluent agissent comme une application entre la prédiction de l'algorithme et une fonction-objectif. La figure 3.1 schématise les notions de fonction boîte noire, de fonction-objectif ainsi que l'algorithme d'optimisation qui suggère un nouvel ensemble d'hyper-paramètres.

La nature stochastique des filtres à particules peut être considérée comme un bruit inclus dans le résultat final. Les mêmes valeurs d'hyper-paramètres ne mènent pas nécessairement à la même performance. Les algorithmes d'optimisation stochastique des hyper-paramètres collectent, à chaque itération, des informations à partir des observations de l'exécution de la fonction boîte-noire et du résultat de la fonction-objectif. À partir de ces informations ils construisent et mettent à jour un modèle aussi proche que possible de cette fonction boîte noire inconnue [Shahriari 2015].

Ce processus est itératif et la méthode choisit soigneusement les prochains paramètres à évaluer. Un algorithme intelligent devrait présenter un bon compromis entre l'exploration de l'espace hyper-paramétrique et l'exploitation de sa modélisation de la fonction-objectif. Dans la phase d'exploration de l'espace, l'évaluation de la fonction-objectif est incertaine. Alors que dans la phase d'exploitation il s'agit d'essayer des valeurs d'hyper-paramètres pour lesquelles nous nous attendons, grâce à modélisation de la fonction boîte-noire, à ce que la fonction-objectif prenne des valeurs élevées si nous recherchons à maximiser.

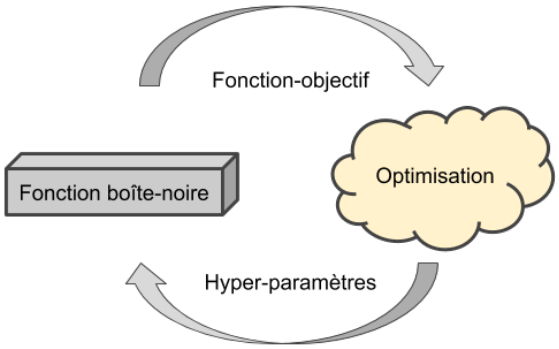


FIGURE 3.1 – Optimisation des hyper-paramètres. Ici la fonction boîte-noire représente une modalité vision et la fonction-objectif représente les métriques qui permettent d'en évaluer les performances.

Si tel est le cas, le modèle de la fonction-objectif est donc proche de la vraie fonction et l'algorithme convergera plus rapidement. Ce qui nécessitera moins d'évaluations de la fonction boîte noire et donc moins de temps de calcul. Dans la littérature, il existe plusieurs propositions dans ce but, mais seules quelques-unes fournissent un code source libre et nous les privilégions ici car l'objectif est de comparer les outils existants et non de développer un nouvel outil.

3.2.1 Optimisation de base : MCMC Metropolis-Hastings

Au début, il peut sembler que l'utilisation d'un algorithme d'optimisation sophistiqué soit difficile à appréhender. Peut-être qu'une méthode basique est suffisante pour trouver des paramètres optimums. Pour savoir si c'est le cas, nous étudions et implémentons un algorithme d'optimisation stochastique classique comme référence de base (*benchmark*).

En général, les méthodes classiques ne reposent pas sur l'estimation de la fonction boîte noire via un modèle et disposent de solides bases théoriques car étudiées depuis longtemps [Minton 1992]. Celles-ci sont relativement simples et faciles à mettre en œuvre. Puisque nous considérons notre fonction comme étant une boîte noire nous avons besoin d'une stratégie qui ne repose ni sur des gradients, ni sur des heuristiques. Ainsi, nous privilégions une optimisation stochastique basée sur les méthodes de Monte-Carlo par chaînes de Markov (MCMC) et utilisons cette méthode comme référence. Nous comparons cette référence, en termes de performance et de facilité d'utilisation à des techniques plus récentes, robustes et libres. MCMC est un ensemble de méthodes stochastiques basées sur une procédure d'échantillonnage. Il est largement utilisé dans la littérature, le plus souvent pour déterminer une approximation d'une loi de probabilité inconnue. Dans notre cas, nous cherchons à obtenir la valeur optimale retournée par la fonction-objectif à la suite de l'exécution de la fonction boîte-noire à partir d'un certain jeu de paramètres qui est inhérent à sa modélisation.

Nous pouvons trouver plusieurs propositions de MCMC dans la littérature, mais nous nous concentrons sur l'algorithme Metropolis Hastings (MH) [Metropolis 1953, BoussaïD 2013].

MH se rapproche d'une distribution à partir de laquelle l'échantillonnage direct est difficile ou impossible (ici notre fonction-objectif) en prélevant des échantillons d'une distribution probabiliste connue au préalable. Il est très performant pour estimer des distributions de grande dimension. D'un point de vue des problèmes d'optimisation, le but est de trouver le minimum d'une fonction-objectif donnée. Dans les cas où cette fonction est difficile à estimer les méthodes itératives MH peuvent se retrouver piégées sur un minimum local. C'est pour éviter que de telles situations surviennent qu'a été proposé en 1983 par Kirkpatrick [Kirkpatrick 1983] une variante appelée "recuit simulé" que nous avons implémentée.

Soit $f(\lambda)$ la fonction qui évalue les hyper-paramètres λ. Nous choisissons un ensemble de paramètres initiaux λ_0 de manière arbitraire, et une densité de probabilité $g(.)$ à partir de laquelle nous pouvons facilement tirer de nouveaux échantillons. Dans notre cas, nous définissions $g(\lambda_t)$ comme la distribution gaussienne centrée sur notre précédent λ_t avec une variance fixée. À chaque itération, nous tirons un échantillon candidat $\lambda^* \sim g(\lambda_t)$ (voir ligne 6 de l'algorithme 1). Ce candidat est utilisé pour calculer le seuil d'acceptation $\alpha = f(\lambda^*)/f(\lambda_t)$. Le nouveau candidat est accepté s'il améliore la valeur de la fonction ($\alpha > 1$). Sinon, le candidat est accepté avec une certaine probabilité donnée par une distribution uniforme. La température T diminue au fil de la recherche, donc au début la probabilité d'accepter des choix de paramètres qui détériorent les performances est grande puis elle diminue graduellement.

Algorithm 1 Algorithme du recuit simulé

1: Choix des valeurs initiales des paramètres λ_0
2: Choix d'une densité de probabilité $g(.)$
3: $\lambda \leftarrow \lambda_0$
4: Initialisation de la température T
5: **while** n < nombre d'itérations maximal **do**
6: $\quad\lambda^* \leftarrow \lambda + \Delta\lambda$ ▷ Suggestion d'un nouveau candidat
7: $\quad\alpha \leftarrow f(\lambda^*)/f(\lambda)$
8: $\quad$**if** $\alpha \geq 1$ **then**
9: $\quad\quad\lambda \leftarrow \lambda^*$
10: $\quad$**else**
11: $\quad\quad\lambda \leftarrow \lambda^*$ avec une probabilité $p(T, f(\lambda^*), f(\lambda))$
12: $\quad$**end if**
13: $\quad T \leftarrow T - \varepsilon$
14: **end while**

Cette variante du recuit simulé peut d'adapter à de nombreux problèmes d'optimisation et ne demande que peu de paramétrage de la part de l'utilisateur, essentiellement la température et les densité de probabilités.

3.2.2 Optimisation bayésienne : processus gaussien

Ces approches sont dites bayésiennes car elles reposent sur théorème de Bayes qui stipule que la probabilité à posteriori d'un modèle M sachant les observations O est proportionnelle à

la vraisemblance de O sachant M multipliée par la probabilité a priori de M :

$$P(M|O) \propto P(O|M)P(M). \tag{3.1}$$

La section précédente décrit comment le MCMC fournit une approximation de la distribution des fonctions-objectif en utilisant des échantillons séquentiels. De plus, si nous avons suffisamment d'échantillons, nous pouvons supposer, en suivant le théorème central limite, que ceux-ci induisent une distribution gaussienne multivariée. Cette hypothèse sur la distribution de la fonction est connue sous le nom de processus de Gauss [Shahriari 2015]. Certaines propositions d'optimisation sont basées sur cette idée ; citons le travail de Snoek *et al.* [Snoek 2012] appelé Spearmint, qui donne et analyse des considérations pratiques pour améliorer cet algorithme d'optimisation bayésien.

Spearmint [Snoek 2012] est bien connu dans la communauté, ayant plus de 1000 citations au moment de l'étude en 2017 et trois fois plus en 2020, ainsi que des sites web supplémentaires consacrés à l'explication de cet outil. L'objectif de Spearmint est de capturer la dépendance entre les hyper-paramètres λ et la fonction-objectif f. Ceci est réalisé en utilisant une distribution de probabilité $p(f|\lambda)$, qui est modélisée avec le processus de Gauss. Cependant, au lieu de trouver le maximum de probabilité, la proposition "projette" (*marginalize*) le modèle utilisant un échantillonnage par tranches. Cette idée permet d'exécuter de nombreuses évaluations en parallèle, chacune avec une loi marginale différente. Cette capacité de parallélisation permet de réduire le temps de calcul pour l'optimisation des paramètres de certaines méthodes coûteuses telles que pour l'apprentissage profond [Snoek 2015]. Plusieurs approches basées sur un processus de Gauss exploitent ce mécanisme de parallélisation [Haftka 2016]. En outre, l'approche introduit le critère *Expected Improvement* (EI) par acquisition de coût qui permet d'échantillonner non seulement de bons paramètres mais aussi des paramètres rapides à évaluer.

3.2.3 Méthodes d'optimisation bayésiennes structurées

Certaines méthodes tentent de décrire la "diversité" (*manifold*) de l'espace des hyper-paramètres par l'utilisation de modèles, en suivant généralement une stratégie d'optimisation itérative. À chaque itération, les méthodes évaluent une hypothèse de paramètres et mettent à jour le modèle. Après un certain nombre d'itérations, ces méthodes proposent un candidat final, selon leur propre méthodologie, qui offre les meilleures performances. Cette famille de méthodes est appelée "Optimisation Séquentielle Basée sur un Modèle" (*Sequential Model-Based Optimization*, SMBO) [Hutter 2011b]. Ces méthodes sont souvent utilisées lorsque la fonction boîte noire est coûteuse à évaluer en temps de calcul. Nous évaluons deux méthodes de l'état de l'art, qui sont sélectionnées en fonction de leur popularité (nombre élevé de citations), de la disponibilité du code source et de leurs performances annoncées.

3.2.3.1 Sequential Model-based Algorithm Configuration

Un exemple de méthode SMBO est *Sequential Model-based Algorithm Configuration* (SMAC) [Hutter 2011b]. Cette approche, citée 600 fois au moment de l'étude en 2017 et maintenant plus de 1300 fois en 2020, repose sur l'apprentissage automatique. L'outil développé est toujours maintenu avec la sortie récente de la version 3. Cette méthode itérative considère l'ensemble des données collectées lors de la proposition d'un nouvel ensemble de candidats, ce qui mène à de meilleurs estimations. Ici SMAC modélise la probabilité a posteriori $p(f|\lambda)$ comme une distribution gaussienne au moyen de forêts d'arbres décisionnels (*random forest*, RF). Pour chaque arbre une moyenne et une variance empirique sont estimées. Ce modèle, en s'appuyant sur les RF, améliore la performance avec une optimisation discrète, s'adapte aisément aux données et est capable de gérer des fonctions bruitées avec des jeux de paramètres de grande dimension.

Premièrement l'ensemble de paramètres candidats sont échantillonnés à partir d'une distribution uniforme donnée dans intervalle fini. Puis les échantillons sont divisés selon un paramètre choisi au hasard. Ils seront utilisés plus tard pour construire les arbres de décisions. Ce mécanisme de division est répété jusqu'à l'obtention d'un nombre minimum d'échantillons par branches. Les configurations sont évaluées suivant le critère *Expected Improvement* ou EI, et les seules les plus prometteuses sont sélectionnées. Pour finir, la meilleur configuration, celle avec le plus grand EI, est comparée à la précédente. La nouvelle proposition est acceptée si elle améliore la valeur de la fonction-objectif et donc les performances de la fonction boîte noire. S'il y a une amélioration alors l'arbre correspondant est utilisé dans la prochaine itération.

3.2.3.2 The Tree-structured Parzen Estimator [2] (TPE)

TPE est également une approche répandue avec 500 citations au moment de l'étude et maintenant plus de 1600 en 2020. Alors que SMAC et Spearmint modélisent explicitement $p(f|\lambda)$, c'est à dire l'estimation de la valeur de la fonction-objectif sachant les paramètres, TPE modélise $p(\lambda|f)$ et $p(f)$. En effet, TPE modélise la probabilité des paramètres suivant s'ils améliorent $p(\lambda|f \geq f^*)$ ou dégradent $p(\lambda|f \leq f^*)$ l'évaluation de la fonction-objectif selon un certain pourcentage des performances de la fonction-objectif observés jusqu'alors.

En pratique, TPE tire de nouveaux échantillons à chaque itération et décide de l'ensemble à essayer à l'itération suivante. TPE tire des échantillons de λ uniformément dans l'espace de configuration, c'est pourquoi il n'a besoin ni de valeurs initiales ni d'ensemble d'entraînement. Puis les échantillons sont évalués par la fonction-objectif f. L'approche sépare les échantillons en deux groupes selon les scores obtenus. Le premier groupe contient tous les échantillons qui améliorent la performance courante f^*, alors que le second contient les autres. Les deux groupes sont utilisés pour modéliser la vraisemblance : pour le premier groupe un modèle $g(\lambda) = p(\lambda|f \geq f^*)$ et pour le second groupe $l(\lambda) = f \leq f^*$. Les modèles utilisent une estimation par noyau (*1-D Parzen estimator*) pour mesurer la densité des groupes à travers une structure hiérarchique. Le but est de créer de nouveaux candidats qui ont plus de chances de se retrouver dans le premier groupe. Ainsi à chaque itération de nouveaux échantillons de $g(\lambda)$ sont

tirés. Ceux avec la meilleure amélioration sont alors utilisés à l'itération suivante. TPE définit le critère EI comme le ratio entre les modèles des deux groupes : $EI = \frac{l(\lambda)}{g(\lambda)}$.

3.2.4 Outils associés

MCMC est un outil robuste pour estimer une distribution de paramètres et son implémentation est aisée. C'est pourquoi nous l'avons implémenté en C++ en suivant l'algorithme classique Metropolis-Hasting décrit par l'algorithme 1 et par [BoussaïD 2013]. Il reçoit en entrée les mêmes données que les autres outils tels que : les bornes de l'espace de configuration, le nombre d'itérations, une configuration initiale du vecteur de paramètres, la fonction boîte-noire et la fonction objectif. Le nombre d'itérations joue un rôle clé dans le résultat final, un faible nombre peut mener à une sous-estimation de la distribution des paramètres mais un trop grand nombre peut mener à un problème de sur-apprentissage.

Spearmint a été crée par des membres de *the Harvard Intelligent Probabilistic Systems Group* [Snoek 2012]. Le code open-source requiert l'installation de Python et MondoDB. Ce dernier est un système de gestion de bases de données, qui ici est utilisé pour conserver les données collectées au cours des itérations lors de l'optimisation. MongoDB permet à différentes applications (ou *threads* de la même application) à accéder au même ensemble de données simultanément. Cela rend la parallélisation possible, mais également le suivi du processus à partir de de la dernière itération évaluée. Ce qui permet d'analyser les résultats dans le cas où il y aurait un problème avec l'évaluation de la fonction-objectif. La documentation de Spearmint est très limitée, avec seulement une petite explication de la manière de mettre en place une expérience simple. Les exemples inclus donnent une meilleure idée des différents paramètres disponibles. Spearmint requiert deux fichiers : un fichier de paramètres qui définit l'espace de configuration des hyper-paramètres, et un fichier Python qui lance la fonction boîte noire. Des paramètres additionnels tel que le nombre d'itérations peuvent être passés via le terminal.

SMAC est un outil librement disponible pour l'optimisation des paramètres de configuration. Il est en développement depuis plusieurs années. La version stable SMAC v2 a été publiée en 2015, c'est cette version que nous avons testée. Depuis 2018, la version v3 a été publiée. SMAC repose sur Java 4, il est facile de l'installer et de le lancer sur différents environnements. Les auteurs fournissent une documentation qui comprend : un guide d'installation rapide et un manuel détaillé de l'outil. L'ensemble est facile à utiliser, il faut définir deux fichiers : un fichier scénario en Python et un fichier de configuration des paramètres. Le premier fichier contient les informations de lancement de la fonction boîte-noire, le chemin vers le fichier de paramètres, et des options d'optimisation (par exemple le nombre d'itérations). Le fichier des paramètres décrit les limites de l'espace de configuration, les valeurs initiales et le type des paramètres.

TPE est également un outil d'optimisation ; il fait parti des bibliothèques Hyperopt et Optunity. Cette dernière supporte différents langages tels que : Python, MATLAB, Octave, R, Julia et

Java. Hyperopt propose une implémentation en Python de TPE et peut être lancé en séquence ou en parallèle via MongoDB. Les deux bibliothèques proposent une documentation basique et de petits exemples d'utilisation. Sa configuration est plus simple que SMAC. La fonction d'optimisation reçoit en paramètres la fonction boîte-noire, le nombre d'itérations et les limites de l'espace de configuration des paramètres.

Le choix de ces différents outils s'est avéré pertinent car nous pouvons observer au travers de l'augmentation du nombre de citations pour chacun d'eux qu'ils ont continué à être utilisés entre le moment de l'étude en 2017 et aujourd'hui en 2020.

3.3 Étude de cas : un algorithme de suivi par détection

La performance de certains algorithmes est dépendante du choix de certains paramètres. Il est difficile de trouver une configuration proche de l'optimum, surtout dans le cas de fonctions stochastiques et cela même pour des utilisateurs expérimentés. Heureusement, de telles configurations peuvent être calculées en ayant recours à des outils d'optimisation. Ce travail s'inscrit dans un projet qui a financé le début de ma thèse : le projet DGA RAPID SERVAT (pour *Suivi Et Reconnaissance Visuels, Adaptatifs, Temps-réel*). Comme exigé par ce projet, nous avons retenu un algorithme de suivi de cibles pour la réalisation de cette étude des outils d'optimisation. Le suivi multi-objet est largement étudié dans la littérature car cela peut-être appliqué dans divers secteurs : la sécurité, les interactions homme-robot parmi d'autres. De part la nature stochastique de l'algorithme de suivi choisi ainsi que le nombre de paramètres qui y sont associés, nous utiliserons ce retour d'expérience sur ces outils pour le réglage (*tuning*) des hyper-paramètres à nos fonctions de reconnaissance d'action développées par la suite et constituant le cœur de la thèse.

3.3.1 Le suivi multi-cibles

L'analyse de ces outils de réglage peut-être effectuée avec n'importe quelle problématique qui peut être formulée comme une fonction-objectif à maximiser ou à minimiser. Néanmoins, la pertinence d'utiliser de tels outils est plus probante lorsque l'on est en présence de méthodes complexes, telle que les approches stochastiques, avec de nombreux paramètres. Certaines approches de suivi multi-objets (*Multi-Object Tracking* MOT) présentent ces caractéristiques [Kim 2006].

Dans ce contexte, le but est suivre la trajectoire des cibles , ici des personnes, au sein d'une vidéo. Les challenges associés sont bien connus : occultations entre cibles, environnement encombré et non contrôlé, trajectoires croisées, etc. Pour surmonter ces défis, des approches ont été proposées et elles peuvent être classées en tant que déterministes ou stochastiques [Smeulders 2013, Watada 2010]. Les paramètres des approches stochastiques sont plus difficiles à régler car le même ensemble de paramètres peut mener à différents résultats à chaque évaluation. Dans la suite la fonction boîte noire considérée est un algorithme de suivi de personnes par détection dans le plan image d'une caméra.

En parallèle, la communauté Vision a créé un ensemble de mesures, par exemple CLEAR-MOT [Bernardin 2008] pour évaluer les performances des systèmes de suivi et de comparer différentes approches selon les mêmes critères. Ces métriques constituent notre fonction-objectif que l'on cherche à optimiser, dans ce cas à maximiser.

3.3.2 Le filtre à particules et ses hyper-paramètres

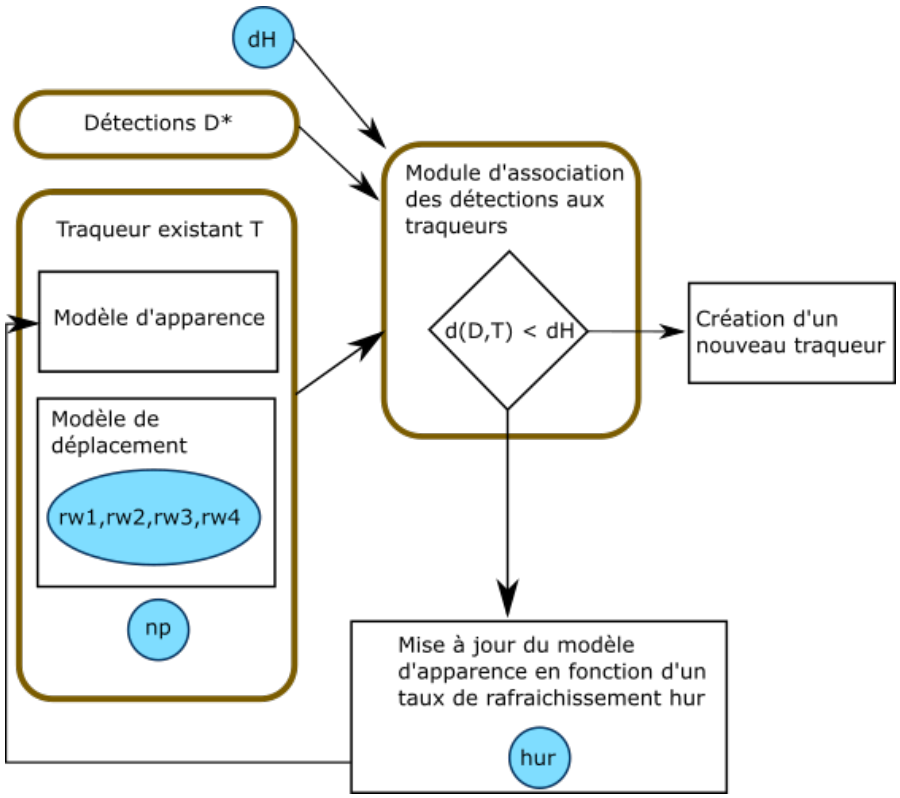


FIGURE 3.2 – Représentation du filtre à particules et des module d'association des détections aux traqueurs dans lesquels apparaissent les hyper-paramètres ici représentés en bleu.

À partir de ces observations, nous menons notre étude dans le cadre du suivi de piétons dans un environnement extérieur. Nous avons donc un certains nombre de paramètres à optimiser par rapport à la fonction-objectif retenue : une métrique définie par CLEAR-MOT, en particulier le critère MOTA. La suite de cette section présente succinctement l'algorithme de suivi retenu comme fonction boîte-noire afin de répertorier les différents hyper-paramètres.

Un challenge public est lié à cette problématique, le MOTChallenge, répertorie et classe les méthodes de l'état de l'art par rapport à leurs performances. En général les résultats sont si proches que même une amélioration de 2 points de pourcentage (pp) du critère MOTA peut mener à gagner 10 places dans le classement. La plupart des approches de suivi que l'on y trouve sont des approches de suivi via suivi-par-détection. Ainsi, nous privilégions une stratégie

classique inspirée de [Breitenstein 2010] au moyen de filtres à particules décentralisés. Dans cette étude nous cherchons à quantifier l'impact d'un bon paramétrage et non des performances en absolue. Dans notre implémentation de suivi-par-détection, l'état de chaque cible est estimé par un filtre à particule (FP) dans le plan image.

Nous utilisons les détections pré-calculées qui sont fournies par le MOTChallenge, ainsi les paramètres liés au détecteur ne sont pas pris en compte dans la phase de paramétrage.

Chaque détection est associée à un traqueur si la distance entre eux est plus petite qu'un seuil dH sinon un nouveau traqueur est créé. En pratique, fixer ce seuil dH n'est pas intuitif, car cela dépend de la taille de l'image, de la fréquence d'image et de la distance entre la caméra et les cibles. Si la valeur est trop petite certaines associations seraient omises alors que si la valeur est trop grande cela mène à des associations erronées. Nous devons donc paramétrer ce seuil dH. Notre modèle d'apparence des traqueurs se construit à partir des distributions couleurs dans les boîtes englobantes des détections représentées par un histogramme H_d. Le modèle d'apparence incorpore les données des histogrammes de couleurs des détections nouvellement associées à ce traqueur avec un certain taux de rafraîchissement hur. L'histogramme H_r du modèle d'apparence est mis à jour suivant la somme pondérée suivante : :

$$H_r = hur * H_r + (1 - hur) * H_d, \tag{3.2}$$

avec 0 < hur < 1 le taux de rafraîchissement de l'histogramme de référence de la cible. Si l'apparence de la cible change peu entre images successives, hur peut prendre une valeur proche de 1. Si la valeur de hur est trop faible il y a un risque d'ajouter des données incorrectes dans le modèle d'apparence. Il est essentiel de trouver une bonne valeur pour hur pour maximiser les performances. Comme nous pouvons le voir sur la figure 3.2, le modèle d'apparence des cibles au sein du traqueur est influencé par ces deux paramètres : dH et hur.

Le modèle de déplacement associé au filtre à particules est un modèle de marche aléatoire. Ainsi l'état est mis à jour de la sorte :

$$X_t = X_{t-1} + \eta_t, \tag{3.3}$$

où η_t est un bruit gaussien indépendant pour chaque paramètre ($\eta_t^* \sim N(0, rwn_*)$). Le bruit de marche aléatoire rwn^* permet au traqueur de se déplacer dans différentes directions à différentes vitesses. Il dépend donc du comportement des cibles. Les valeurs faibles sont idéales pour suivre les piétons dont la vitesse apparente est plus lente alors qu'une valeur plus élevée permet de suivre les cibles dont la vitesse apparente est plus rapide. Ces vitesses apparentes sont dépendantes de la configuration spatiale de la caméra par rapport à la scène. Les séquences contiennent souvent les deux cas et régler la valeur de ce paramètre n'est pas intuitif. Comme nous pouvons le voir sur la figure 3.2, le modèle de déplacement des cibles au sein du traqueur est influencé par ces quatre paramètres : $nw1,\ nw2,\ nw3,\ nw4$.

Notre implémentation du filtre à particules applique un ré-échantillonage des particules à chaque itération. Le nombre de particules np joue un rôle important. Dans de nombreux cas, un grand nombre de particules peut améliorer l'estimation de la position des différents piétons au

FIGURE 3.3 – Exemples de trames issues de la séquence S2L1 du jeu de données PETS 2009.

détriment du coût CPU. Parfois un faible nombre de particules est suffisant pour les scénarios les plus simples c.à.d avec peu de cibles. Ainsi nous avons besoin de trouver le nombre de particules avec le meilleur ratio temps de calcul et justesse.

Pour résumer, notre implémentation de traqueur MOT suivi-par-détection nécessite le paramétrage de 7 paramètres : le seuil de distance d'association dH, le taux de rafraîchissement de l'histogramme hur des modèles d'apparence des traqueurs, les 4 paramètres pour le bruit de la marche aléatoire rwn pour le modèle de déplacement des traqueurs et le nombre de particules np. Ces composantes sont mises ensemble dans un vecteur $\lambda = \{dH, hur, rwn1, rwn2, rwn3, rwn4, np\}$, qui décrit l'ensemble des paramètres utilisés dans les processus d'optimisation.

3.4 Évaluations

Dans cette section nous présentons les jeux de données sur lesquels les paramètres de l'algorithme de suivi seront évalués et optimisés. Nous décrivons également le protocole d'évaluation des différents outils. J'ai développé le framework pour l'évaluation de ces outils ; celles-ci ont été étendues par la suite en concertation avec un post-doctorant de l'équipe : Francisco Madrigal.

3.4.1 Description des jeux de données

Nos évaluations sont effectuées sur deux séquences extraites de deux jeux de données publics : PETS 2009 [Ferryman 2009] et ETH [Ess 2008]. Ces deux jeux de données présentent différents challenges pour les systèmes de suivi et permettent de vérifier la reproductibilité de notre étude.

Le jeu de données PETS 2009 propose des séquences diversifiées, allant du suivi d'un unique piéton à de l'estimation de foule. La densité varie de faible à très dense. Nous utilisons la séquence S2L1, voir figure 3.3 qui consiste en 8 séquences synchronisées observant une même scène extérieure. Chaque vidéo a 795 trames enregistrées à 7 trames par secondes avec une résolution de 640x480 pixels. La séquence est de densité moyenne, avec 19 piétons et est destinée au suivi de multi personnes avec des situations d'occultations. Parmi l'ensemble des caméras qui

FIGURE 3.4 – Exemples de trames issues de la séquence *Sunny Day* du jeu de données ETH.

filment la scène, nous utilisons la caméra fixe numéro 2, car c'est cette séquence en particulier qui a été sur-exploitée dans la littérature. Les résultats obtenus ont presque atteint la limite et il n'y a qu'une petite amélioration entre les différentes approches. Nous voulons montrer que le classement d'un même système de suivi peut changer avec un réglage optimisé de ses hyper-paramètres.

Le jeu de données ETH regroupe 8 séquences enregistrées par un système de paire de caméras montées sur une poussette. Les caméras ont une résolution de 640x480 pixels avec 13 à 14 trames par seconde. Chaque vidéo montre différents endroits d'une rue animée. La caméra est mobile, se déplace et filme la scène avec un point de vue bas et non surélevé. Ce sont des séquences difficiles surtout à cause de la position de la caméra qui augmente le nombre de situations d'occultations. Les outils sont évalués sur la séquence *Sunny Day* qui contient 338 trames, voir la figure 3.4.

Ces deux séquences représentent des scénarios classiques dans le suivi de piétons, d'un part avec une caméra statique située à distance et en hauteur qui est idéale pour des fins de surveillance, et l'autre avec la caméra mobile qui est idéale pour étudier les interactions dans un environnement. Nous menons les évaluations en utilisant le kit d'évaluation et la vérité terrain fournis par le MOTChallenge.

3.4.2 Protocole expérimental

Nous analysons les performances des outils précités d'optimisation des hyper-paramètres. La fonction boîte-noire considérée est est donc notre traqueur MOT et la fonction-objectif est une mesure de justesse définie par CLEAR-MOT. La performance absolue de ce traqueur n'est pas centrale, il existe certes des traqueurs plus performants, mais nous étudions les performances relatives par rapports aux divers outils d'optimisation. Notre fonction boîte noire prend en entrée l'ensemble des paramètres λ et retourne une valeur de performance : le MOTA.

3.4.2.1 Configuration des outils

Les trois outils évalués requièrent la définition d'un espace de recherche, soit au moyen d'un intervalle, ou d'une covariance. De plus SMAC et MCMC demandent de choisir des valeurs

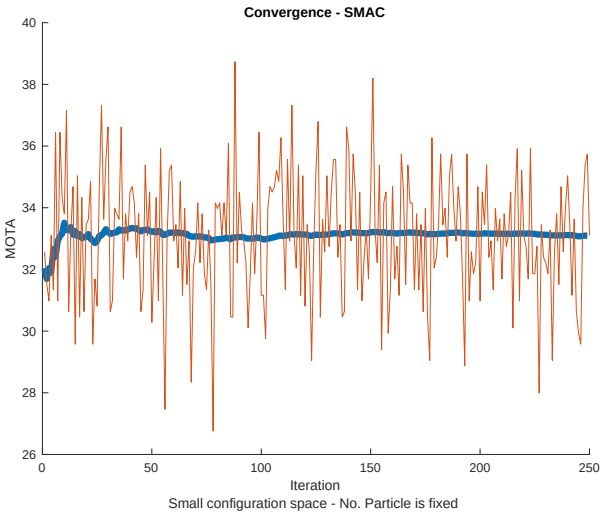
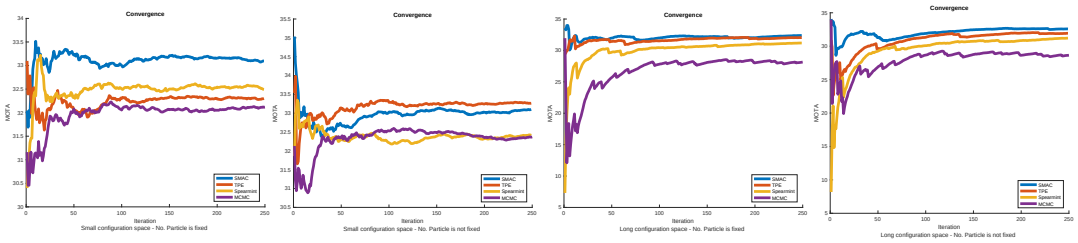


FIGURE 3.5 – Évaluation de SMAC en utilisant la configuration SF. Les évaluations de la fonction, en utilisant la métrique MOTA, sont affichées en orange. La distribution cumulée est affichée en bleu. L'axe des abscisses (resp. ordonnées) représente les itérations (resp. les valeurs du MOTA).

initiales. Pour ces valeurs initiales nous avons choisi la valeur au milieu de l'intervalle. Les outils sont capables d'appeler notre fonction boîte-noire avec des paramètres à tester qui se situent dans ces intervalles de recherche.

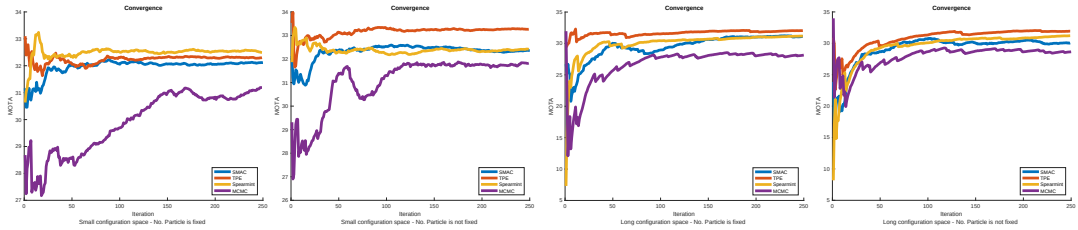
Nous avons besoin de définir un fichier de configuration pour l'usage de ces outils. En général trois aspects sont préciser : l'intervalle de recherche, les valeurs initiales et le type de variable. Comme mentionné dans la section 3.3.2, nous avons λ avec 7 variables. Parmi elles, seul le taux de mise à jour de l'histogramme de référence hur est bien délimité, car il est normalisé entre 0 et 1. Pour les autres, nous ne pouvons utiliser que notre expérience pour fixer de manière empirique ces limites.

Ainsi, nous définissons un petit espace de configuration S fixé autour de là où nous pensons trouver la configuration optimale. Dans ce cas S est défini comme suit : S $= \{dH \in [0.1, 10], hur \in [0.1, 1], rwn* \in [0.1, 3], np \in [10, 100]\}$. Cependant nous ne savons pas si ces paramètres sont les meilleurs, donc nous définissons un espace de recherche plus large L comme suit : L $= \{dH \in [0.1, 20], hur \in [0.1, 1], rwn* \in [0.1, 20], np \in [10, 100]\}$. Aussi il pourrait y avoir une corrélation entre le nombre de particules np et les autres paramètres qui influenceraient le résultat final. En effet un nombre élevé de particules pourrait compenser les limites des autres paramètres. Nous analysons l'influence du nombre de particules au travers de deux expériences notée par la suite F ou NF suivant si le nombre de particules est fixé ou non.



(a) SF (b) SNF (c) LF (d) LNF

FIGURE 3.6 – Évaluation de la vitesse de convergence. Distribution cumulée du MOTA de SMAC [Hutter 2011b], TPE [Bergstra 2011], MCMC et Spearmint [Snoek 2012] sur la séquence PETS 2009-S2L1.



(a) SF (b) SNF (c) LF (d) LNF

FIGURE 3.7 – Évaluation de la vitesse de convergence. Distribution cumulée du MOTA de SMAC [Hutter 2011b], TPE [Bergstra 2011], MCMC et Spearmint [Snoek 2012] sur la séquence *Sunny Day* de ETH.

3.4.2.2 Évaluations et protocole de comparaison

Dans la littérature, il existe différentes mesures pour évaluer les performances des approches de suivi. Les plus connues sont les mesures CLEAR-MOT qui sont par ailleurs implémentées dans le kit de développement du MOTChallenge. Parmi toutes les mesures proposées par le CLEAR-MOT, nous retenons la mesure de justesse du suivi multi-objets (MOTA) en tant que fonction-objectif que l'on va chercher à maximiser. C'est l'une des mesures clé pour comparer différentes approches de suivi, en effet le MOTChallenge classe par défaut les approches par MOTA décroissant. Le MOTA est défini comme suit :

$$\text{MOTA} = 1 - \frac{\sum_t (m_t + fp_t + mme_t)}{\sum_t g_t}, \tag{3.4}$$

avec m_t le nombre de cible manquées, fp le nombre de faux positifs, et mme le nombre de mauvaises associations, et g_t le nombre de cibles, le tout au temps t. C'est donc notre fonction-objectif que l'on chercher à maximiser.

Les résultats du traqueur MOT sont évalués vis-à-vis de la vérité terrain. La mesure combine les informations de détections manquées, d'incompatibilité entre les détections et les faux positifs. Quand nous comparons les meilleures approches du 2D MOTChallenge 2015, nous

observons que les approches atteignent une précision de suivi multi-objets (MOTP) similaire. L'écart-type est de 0.9 %. Le MOTP correspond à l'erreur totale de position estimée pour les paires cibles-hypothèse appariées sur toutes les trames, la moyenne est calculée par le nombre total de correspondances effectuées. Elle montre la capacité du traqueur à estimer des positions précises des cibles, indépendamment de son aptitude à conserver des trajectoires cohérentes. Néanmoins la différence se situe dans la justesse (MOTA) où il y a un écart-type de 5 %. La mesure cruciale pour départager les différentes approches est donc le MOTA et c'est pourquoi cette mesure est retenue en tant que fonction-objectif.

Nous menons les évaluations sur les séquences vidéo qui sont divisées en deux ensembles. Un ensemble pour l'entraînement qui est utilisé pour la recherche des paramètres avec les outils d'optimisation. Les paramètres ainsi trouvés sont ensuite évalués sur le reste de la vidéo qui constitue l'ensemble de test. Cette division permet d'éviter le sur-apprentissage et de booster les configurations qui généralisent le mieux. En raison de la nature stochastique de notre traqueur MOT, nous évaluons dix fois chaque configuration possible de chacun des outils. Les résultats ainsi présentés sont la moyenne de toutes les évaluations. Cela permet également d'évaluer la stabilité de ces outils et leur vitesse de convergence. Ce dernier est un facteur important qui peut mener à favoriser un outil par rapport à un autre.

Toutes les évaluations ont été effectuées sur un Dell Precision Tower 3620, avec un Intel Xeon CPU v5 de 3.60 GHz et 8 cœurs, 16 GB de RAM sur un système Linux (Ubuntu 14.04).

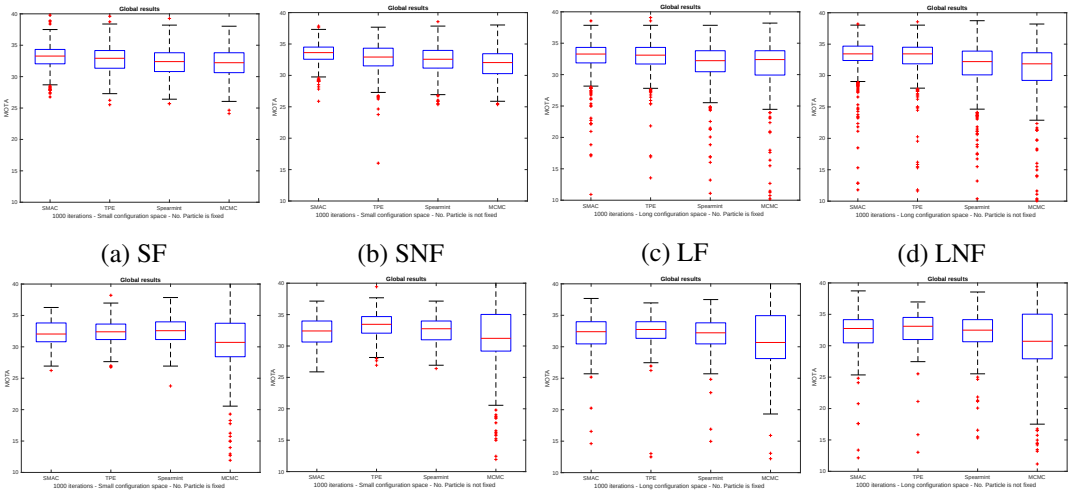
Traditionnellement dans le domaine de l'optimisation, un algorithme converge une fois qu'il a atteint une valeur stationnaire pour la fonction-objectif. Ce n'est pas possible lors de l'optimisation des méthodes stochastiques. Dans la littérature il y a de nombreux critères de terminaison mais la sélection de ceux qui sont les plus appropriés dépendent de la fonction à évaluer. C'est pourquoi, au lieu d'utiliser un critère de convergence automatique, les outils d'optimisation s'arrêtent après un certain nombre d'itérations fixé a priori. En théorie, les méthodes convergent vers la valeur optimale quand le nombre d'itérations tend vers l'infini.

De façon à analyser ces outils selon un critère de convergence, nous avons retenu un critère de convergence faible qui estime la distribution cumulée des paramètres mesurés. Nous pouvons observer un exemple dans la figure 3.5 à partir des résultats de SMAC. La ligne orange montre les évaluations du traqueur MOT, en termes de MOTA, et la ligne bleue correspond à la distribution cumulée. Après 50 itérations, SMAC commence à se stabiliser. Cela signifie que la plupart du temps SMAC explore des zones de l'espace de paramètres qui mènent à des valeurs de MOTA hautes (33).

3.4.3 Résultats

Nous présentons les résultats obtenus pour chaque outil d'optimisation des hyper-paramètres sur les deux séquences vidéos. Nous pouvons donc observer leurs comportements dans deux conditions différentes, caméra statique ou dynamique, afin de voir leur capacité d'adaptation.

Nous analysons les outils suivants 7 critères : (1) la vitesse de convergence, (2) la stabilité de la valeur optimale, (3) la valeur de la mesure MOTA, (4) le temps de calcul, (5) l'influence du choix des paramètres initiaux, (6) la taille de l'ensemble d'entraînement et (7) le nombre de



(a) SF (b) SNF (c) LF (d) LNF

(e) SF (f) SNF (g) LF (h) LNF

FIGURE 3.8 – Stabilité des performances. Analyse de résultats sur la séquence S2L1 de PETS en haut, et sur *Sunny Day* de ETH en bas. La distribution des valeurs de MOTA, obtenue avec 1000 itérations pour chaque outil, est affichée. De gauche à droite : SMAC [Hutter 2011b], TPE [Bergstra 2011], Spearmint [Snoek 2012] et MCMC.

particules choisi. À partir de l'ensemble de ces évaluations nous proposons un classement des outils dans le tableau 3.9.

Vitesse de convergence Nous analysons le critère de convergence faible de chacun des outils sous différentes configurations : un espace d'exploration restreint S et un plus large L, avec un nombre de particules fixé F ou non-fixé NF. Les figures 3.6 et 3.7 présentent les résultats. Nous observons que les outils SMAC et TPE se stabilisent plus rapidement à des valeurs plus élevées que les autres. La performance de Spearmint se situe dans la moyenne dans la plupart des cas.

Stabilité Nous analysons la stabilité de la performance des outils dans les 4 configurations SF (espace S avec nombre de particules fixé), SNF (espace S avec nombre de particules variable), LF (espace L avec nombre de particules fixé), LNF (espace L avec nombre de particules variable). La figure 3.8 présente les résultats de ses évaluations menées avec 1000 itérations, ce qui signifie que chaque outil évalue 1000 fois la fonction-objectif. Le MOTA est utilisé pour construire des boîtes à moustaches qui représentent la distribution de la performance MOTA. Les deux figures indiquent la valeur médiane (ligne rouge) et le premier et le troisième quantile (boîte bleue) et les valeurs aberrantes (croix rouges). Nous y observons que les boîtes de l'outil MCMC sont plus grandes que toutes les autres.

Justesse Les résultats précédents indiquent que TPE et SMAC ont un meilleur taux de convergence avec une utilisation efficace de chaque itération. Les tableaux 3.1 et 3.2 présentent les

TABLE 3.1 – Évaluation du MOTA sur la séquence S2L1 du jeu de données PETS 2009.

		MCMC		SMAC		TPE		Spearmint	
Conf.	It.	Ent.	Test	Ent.	Test	Ent.	Test	Ent.	Test
SF	250	32.75	21.65	33.45	**20.70**	38.20	**21.12**	33.47	21.94
	500	30.28	21.39	**33.55**	20.58	39.44	20.58	33.53	20.30
	1000	33.10	20.64	33.46	20.55	39.61	20.88	33.46	**21.96**
	2000	**33.63**	**21.97**	33.43	20.51	**39.79**	20.78	**34.01**	21.62
SNF	250	**35.04**	20.95	**33.97**	21.31	39.44	**21.67**	33.66	18.78
	500	31.69	19.89	33.87	**22.56**	39.79	20.32	**33.99**	21.86
	1000	30.99	18.72	33.80	21.76	37.68	21.45	33.96	20.53
	2000	34.33	**22.59**	33.70	21.74	**40.14**	20.21	33.94	**22.21**
LF	250	34.51	21.05	33.61	21.99	36.97	**22.55**	33.87	22.12
	500	33.98	19.02	33.28	22.96	38.73	21.77	34.03	**22.56**
	1000	**35.56**	18.82	33.56	21.75	39.09	21.62	33.91	22.28
	2000	33.45	**22.34**	**33.84**	**22.16**	**39.09**	21.51	**34.23**	19.22
LNF	250	32.75	21.18	33.95	20.85	36.97	21.52	34.26	5.43
	500	30.46	20.79	33.92	20.55	37.32	21.40	34.27	20.76
	1000	**33.45**	20.75	33.99	**21.85**	38.56	21.41	34.25	**22.00**
	2000	31.87	**22.06**	**34.01**	21.79	**40.14**	**21.97**	**34.51**	21.11
Écart-type		1.53	1.17	0.23	0.75	1.05	0.60	0.30	3.98

résultats en termes de précision en utilisant la métrique MOTA. L'ensemble des outils sont évalués pour chacune des configurations avec un nombre fixe d'itérations en utilisant l'ensemble d'entraînement (la première partie de la séquence). Au final, l'outil nous donne une valeur de MOTA et un paramétrage des hyper-paramètres. Les valeurs des hyper-paramètres sont évaluées 10 fois pour chaque outil et nous montrons la valeur moyenne dans la colonne "Entrain." (pour entraînement). Dans l'ensemble des cas la variance est inférieure à 0.2. Puis ces paramétrages sont testés 30 fois sur l'ensemble de test. Les résultats sont disponibles dans la colonne "Test". La dernière ligne indique l'écart-type de chaque colonne.

Analysons maintenant les séquences entières en utilisant les meilleurs paramétrages trouvés par les outils. Les résultats sont disponibles sur les tableaux 3.3 et 3.4. Dans les deux cas, SMAC et TPE obtiennent les meilleurs résultats. Les résultats présentés sur la page web du MOTChallenge sont les moyennes des approches de suivi sur l'ensemble des séquences du jeu de données. Néanmoins il est possible d'afficher les résultats individuels pour chacune des séquences vidéo. Nous avons calculé les moyennes et écarts-types des résultats présentés à cette date (2017), et ces résultats sont présentés sur les deux dernières lignes des tableaux.

Temps de calcul Le tableau 3.5 et la figure 3.9 montrent les moyennes des temps de calcul nécessaires pour chaque configuration (S, L) en utilisant la séquence vidéo S2L1 de PETS 2009. Le temps d'exécution de chaque itération de MCMC est pratiquement le même, avec une aug-

TABLE 3.2 – Évaluation du MOTA sur la séquence *Sunny Day* du jeu de données ETH.

Conf.	It	MCMC		SMAC		TPE		Spearmint	
		Ent.	Test	Ent.	Test	Ent.	Test	Ent.	Test
SF	250	**42.39**	40.75	34.39	40.70	46.95	40.71	35.00	40.62
	500	11.68	29.91	35.16	40.84	**48.48**	40.62	**36.35**	40.65
	1000	41.12	**40.79**	35.25	40.88	46.70	**40.95**	35.98	40.64
	2000	25.13	35.09	**35.71**	**40.97**	48.22	40.69	36.23	**40.65**
SNF	250	**43.91**	40.72	34.96	40.62	45.43	**41.03**	36.56	40.52
	500	12.69	39.83	33.73	40.77	**48.22**	40.66	36.34	**40.81**
	1000	29.44	40.74	34.47	40.80	47.21	40.75	**37.03**	40.44
	2000	20.05	**40.75**	**35.17**	**40.85**	47.21	40.59	35.86	40.60
LF	250	**43.91**	40.48	34.41	40.66	47.21	40.77	35.04	40.78
	500	23.86	13.17	**35.58**	40.66	47.21	**41.01**	35.04	**41.21**
	1000	27.16	**40.68**	35.30	40.73	47.46	40.50	37.08	40.84
	2000	14.98	34.04	34.66	**40.81**	**47.62**	40.67	**37.97**	40.58
LNF	250	**31.22**	**40.88**	35.33	40.89	46.45	40.79	35.92	**41.02**
	500	14.72	36.34	33.75	40.72	**49.49**	**40.82**	36.05	40.23
	1000	31.73	40.66	35.22	40.75	47.46	40.80	36.38	40.57
	2000	22.59	40.22	**35.90**	**40.89**	48.73	40.75	**37.73**	36.30
Écart-type		10.81	6.95	0.62	0.10	0.94	0.14	0.84	1.08

mentation linéaire par rapport au nombre d'itérations. Alors que SMAC est un peu plus lent que MCMC, il est plus rapide que les autres outils. Il est à noter que le temps d'exécution de Spearmint croit à chaque itération, ce qui peut constituer un frein. De notre expérience, nous observons que Spearmint obtient un meilleur résultat avec plus d'itérations. Les résultats sur la séquence ETH présentés sont similaires.

Influence des valeurs initiales Lors de l'optimisation, le choix des paramètres initiaux peut influencer le résultat final. TPE et Spearmint utilisent les bornes inférieures des intervalles définies par les configurations S et L comme point de départ par défaut et ce ne peut pas être modifié. Cependant, SMAC et MCMC permettent de choisir une valeur initiale. Nous évaluons 10 ensembles de paramètres initiaux générés aléatoirement 5 fois. Nous avons utilisé la configuration qui permet une exploration plus large des espaces de paramètres, L, sans fixer le nombre de particules car c'est dans cette configuration que nous obtenons les meilleures performances. Les outils sont évalués 30 fois en utilisant l'ensemble de test. Les moyennes et les écarts-types du MOTA obtenus sont affichés dans le tableau 3.6 pour les séquences issues de PETS 2009 et ETH.

Influence de la taille de l'ensemble d'entraînement Les performances des outils d'optimisation s'améliorent plus il y a de données disponibles. Cependant ce n'est pas toujours possible. Nous analysons comment la taille de l'ensemble d'entraînement influence les résultats. Nous considérons 4 tailles correspondant au 5, 10, 15 et 20 premiers pourcent de la vidéo. Le

TABLE 3.3 – Évaluation sur toute la séquence S2L1 de PETS 2009.

Méthode	Conf.	Rcll	Prcn	FP	FN	MOTA	MOTP
MCMC	SF	75.07	65.12	1869	1158.60	**33.98**	69.37
	SNF	74.49	65.18	1849	1185.03	33.83	69.55
	LN	74.15	65.03	1853	1201.23	33.39	69.37
	LNF	74.29	65.24	1839	1194.60	33.85	69.47
SMAC	SF	74.88	64.78	1892	1167.50	33.21	69.39
	SNF	75.33	65.24	1865	1146.03	34.26	69.56
	LN	75.11	65.12	1869	1157.03	34.01	69.39
	LNF	75.80	65.60	1847	1125.07	**35.13**	69.61
TPE	SF	75.01	65.01	1876	1160.63	33.68	69.35
	SNF	75.10	64.88	1888	1157.40	33.44	69.52
	LN	75.09	65.15	1866	1157.53	34.05	69.33
	LNF	75.49	65.44	1853	1139.30	**34.70**	69.57
Spearmint	SF	74.76	64.62	1901	1173.47	32.73	69.37
	SNF	75.33	65.21	1867	1146.23	34.21	69.61
	LN	74.90	64.91	1882	1165.90	33.50	69.41
	LNF	75.73	65.33	1857	1141.13	**34.33**	69.61
MOTChallenge	Moyenne	83.29	81.24	1029	775	57.13	71.17
MOTChallenge	Écart-type	5.541	10.77	894	261.054	16.23	0.442

tableau 3.7 montre les résultats obtenus sur les deux séquences. Tous les outils fonctionnent correctement à partir de 10 %. Pour MCMC, SMAC et TPE la performance augmente d'autant plus qu'il y a de trames dans l'ensemble d'entraînement. Quand à Spearmint, il est intéressant de noter qu'il obtient de meilleures performances en test que d'autres outils avec l'utilisation de moins de données.

Influence du nombre de particules Lors des évaluations le nombre de particules est fixé à 30. Cette valeur présente un bon compromis entre le temps de calcul et les performances. Puisque le nombre de particules influence les résultats, nous proposons une évaluation avec différentes valeurs pour la configuration L. Cette analyse n'est effectuée se limite à SMAC et TPE car ce sont les deux outils qui ont montré les meilleurs performances jusqu'alors. Les hyper-paramètres optimisés sont ensuite évalués sur toute la séquence PETS et la moyenne de ces évaluations est reportée sur le tableau 3.8.

3.4.4 Discussion

Les outils d'optimisation des hyper-paramètres sont conçus pour fonctionner avec des algorithmes qui nécessitent le réglage de plusieurs variables. Ils doivent donc être intuitifs à utiliser et bien s'adapter à différents scénarios. Nos évaluations donnent un aperçu original de ces outils et de leur pertinence dans le cadre du MOT afin de guider les utilisateurs dans leur choix.

TABLE 3.4 – Évaluation sur toute la séquence *Sunny Day* de ETH.

Méthode	Conf.	Rcll	Prcn	FP	FN	MOTA	MOTP
MCMC	SF	71.67	85.95	217.40	527	**59.37**	78.19
	SNF	71.16	86.3	209.9	536	59.31	78.19
	LN	71.82	86.03	217.2	524	59.51	78.06
	LNF	71.28	86.1	213.8	534	59.21	77.93
SMAC	SF	71.76	86.14	214.5	525	**59.64**	77.97
	SNF	71.72	86.03	216.4	525	59.46	78.2
	LN	71.46	86.27	211.2	530	59.52	77.9
	LNF	71.72	86.01	216.90	526	59.40	78.06
TPE	SF	71.74	86.16	214.5	525	59.6	78.07
	SNF	71.77	86.25	212.4	525	**59.72**	78.09
	LN	71.6	86.11	214.7	528	59.42	78.07
	LNF	71.78	86.2	213.6	524	59.69	77.92
Spearmint	SF	71.62	85.79	220.6	527	59.17	78.13
	SNF	71.65	86.04	216.2	527	**59.41**	78.07
	LN	71.77	86.25	212.8	525	59.31	78.25
	LNF	71.85	86.48	218.7	523	58.98	78.41
MOTChallenge	Moyenne	61.13	86.26	211.67	736	47.66	77.77
MOTChallenge	Écart-type	3.22	10.40	211.58	63	10.87	1.88

Vitesse de convergence L'analyse des critères de convergence faibles (voir figures 3.6 et 3.7) montre des résultats intéressants. Tant SMAC que TPE trouvent les valeurs les plus élevées du MOTA la plupart du temps. Ils convergent plus rapidement que les autres sur n'importe quelle configuration. Notre référence MCMC prend plus de temps à se stabiliser et la distribution cumulative atteint des valeurs MOTA plus faibles que les autres. Spearmint peut trouver dans la configuration restreinte S des valeurs de MOTA élevées en quelques itérations. Cependant, il faut plus de temps pour traiter la configuration élargie L. En outre, la configuration restreinte S est choisie à l'endroit où elle devrait avoir la configuration optimale des paramètres. Elle est définie sur la base de connaissances expertes sur les paramètres dans le cadre du MOT.

Ces informations a priori induites dans la configuration S permettent de faire converger les outils d'optimisation plus rapidement avec des valeurs MOTA plus stables par rapport à la configuration L. Dans le cas où la connaissance d'un expert par rapport aux paramètres n'est pas disponible, il est possible d'utiliser un espace de configuration plus large et d'obtenir, au prix d'un nombre d'itérations plus grand, des résultats similaires.

Le nombre de particules Le nombre de particules joue un rôle important dans notre approche de suivi. L'intuition initiale de fixer le nombre de particules afin de réduire le nombre de paramètre à optimiser n'a pas apporté de gains. En effet les configurations NF (Non Fixé) convergent plus rapidement que les autres. Les outils utilisent cette variable pour compenser les autres. Plus

TABLE 3.5 – Temps de calcul, en heures, nécessaire aux outils avec la séquence S2L1 du jeu de données PETS 2009.

		Temps (heure)			
Conf.	No. Iter.	MCMC	SMAC	TPE	Spearmint
SF	250	0.73	0.83	1.86	1.14
	500	1.12	1.69	2.96	5.83
	1000	2.79	3.27	7.34	14.78
	2000	5.41	**6.72**	**11.60**	38.36
SNF	250	0.77	1.02	2.02	1.92
	500	1.14	2.26	3.21	4.64
	1000	2.86	4.46	8.00	12.24
	2000	5.37	**9.42**	**13.38**	37.17
LF	250	0.70	0.83	1.85	1.15
	500	1.09	1.69	2.92	5.80
	1000	2.83	3.40	7.31	14.71
	2000	5.33	**6.63**	**11.52**	37.17
LNF	250	0.72	1.15	2.08	1.98
	500	1.08	2.26	3.31	4.99
	1000	2.77	4.60	8.66	12.78
	2000	5.24	**8.73**	**12.57**	39.95

clairement, un grand nombre de particules est favorisé lorsque les autres paramètres sont mal paramétrés pour suivre les personnes. Par exemple, les outils testent de grandes valeurs pour le paramètre np quand le paramètre de bruit de la marche aléatoire est petit. De plus de figer ou ne pas figer ce paramètre change le comportement de TPE en améliorant ses performances dans le cas de la configuration S pour les deux séquences. Ainsi nous concluons que à la fois TPE et SMAC convergent généralement vers la valeur du MOTA optimal plus rapidement que les autres approches. Leur usage est donc recommandé lorsqu'il est nécessaire de n'effectuer que peu d'évaluations de la fonction.

TABLE 3.6 – Évaluation de l'influence des paramètres initiaux. Moyenne du MOTA avec 10 ensembles de paramètres initiaux choisis aléatoirement. Ces paramètres initiaux sont testés avec la configuration L sans avoir fixé le nombre de particules.

		Entraînement		Test	
Séquence	Méthode	MOTA	Écart-type	MOTA	Écart-type
PETS	MCMC	33.95	0.72	19.13	1.56
	SMAC	**34.05**	**0.08**	**22.39**	**0.66**
ETH	MCMC	19.77	6.37	27.92	14.71
	SMAC	**34.81**	**0.76**	**40.23**	**0.30**

TABLE 3.7 – Évaluation de la performance en faisant varier la taille de l'ensemble d'entraînement.

		PETS				ETH			
		5	10	15	20	5	10	15	20
MCMC	Entraînement	31.0	31.1	31.5	31.9	6.8	21.2	23.5	37.2
	Test	17.6	17.6	18.9	19.5	18.2	19.2	39.0	39.7
SMAC	Entraînement	30.7	31.2	32.5	33.8	17.9	34.9	35.4	39.3
	Test	21.4	21.9	22.1	22.2	28.1	38.3	40.2	40.5
TPE	Entraînement	37.9	39.1	37.1	38.2	24.4	48.6	46.4	40.2
	Test	20.1	20.0	22.6	22.8	18.2	19.2	39.0	39.2
Spearmint	Entraînement	7.1	34.3	34.9	32.1	33.1	33.1	33.1	32.8
	Test	18.6	21.2	22.6	20.1	38.5	40.1	39.3	40.0

TABLE 3.8 – Évaluation sur la séquence de PETS 2009 en fixant le nombre de particules à différentes valeurs.

	TPE - Config. LF					SMAC - Config. LF				
Nb. part.	10	30	50	70	100	10	30	50	70	100
Heure	1.94	2.17	2.38	2.69	3.17	1.38	1.69	1.82	2.56	2.66
Rappel	74	75.1	75.5	75	75.7	74.4	75.1	75.5	75.6	75.8
Précision	64.1	65.1	65.6	65.0	65.4	64.7	65.1	65.4	65.5	65.5
FP	1924	1866	1847	1874	1861	1884	1869	1858	1855	1859
FN	1209	1158	1139	1165	1130	1190	1157	1141	1132	1124
MOTA	31.6	34.1	**34.9**	34.6	34.7	33.0	34.0	34.5	34.8	**34.9**
MOTP	69.1	69.3	69.4	69.6	69.7	69.1	69.4	69.5	69.6	69.6
MOTAL	32.6	34.9	35.5	**35.6**	35.6	33.8	34.8	35.4	35.7	**35.8**

Stabilité des outils Idéalement les outils devraient explorer l'espace de configuration de manière efficace, en évaluant des paramètres prometteurs. Lorsque c'est le cas nous nous attendons à ce que la distribution des évaluations de la fonction suive une distribution gaussienne avec une faible variance. Si ce n'est pas le cas, la variance augmente lors de l'évaluation de paramètres sous-optimaux, ce qui se traduit en un gaspillage d'itérations. Les boîtes à moustache quantifient cette analyse sur la figure 3.8. Elles montrent surtout que MCMC est instable et évalue et accepte des hyper-paramètres menant à de faibles valeurs de MOTA. D'un autre côté TPE apparaît comme plus robuste, en faisant un usage avisé de chaque itération afin d'explorer les paramètres les plus prometteurs. SMAC quand à lui, surpasse les autres sur le jeu de données PETS, en présentant une plus faible variance. Il trouve de paramètres stables qui généralise bien ce scénarios filmé avec une caméra statique. Cependant SMAC gère difficilement le cas de la caméra mobile du jeu de donnée ETH. En effet, SMAC essaye d'optimiser rapidement la variable liée à la distance d_H. Il oscille donc entre des configurations adaptée aux instants où les cibles sont éloignées de la caméra et d'autre mieux adaptées lorsque les cibles sont proches de la caméra. C'est pourquoi il est recommandé d'utiliser SMAC ou TPE pour les fonctions qui contiennent du bruit stochastique.

Justesse D'après les tableaux 3.1 et 3.2 les résultats de MCMC sont instables, aboutissant à différentes valeurs à chaque itération. C'est à cause de sa conception, elle ne prend pas en compte les évaluations de fonctions bruitées c'est à dire lorsque deux évaluations des même paramètres en entrée mènent à deux résultats de MOTA différents. Sa performance est affectée par la sortie de nature stochastique du filtre à particule.

Spearmint fournit des résultats plus stables avec un écart type de 0.3. Cependant, les valeurs des paramètres donnés en sortie ne se généralise pas avec le reste de la séquence. De plus le nombre d'itérations requis pour obtenir des résultats est proportionnel à la taille des espaces à explorer défini par les configurations S et L. La configuration L demande plus d'itérations. A la différence de TPE qui malgré une convergence lente, propose des paramètres robustes. Il présente les meilleurs valeurs sur l'ensemble de test car TPE rend compte du meilleur ensemble de paramètres trouvé. SMAC apparaît comme plus stable que les autres à la fois dans les ensembles d'entraînement et de test. Les résultats sur l'ensemble d'entraînement sont généralement cohérents avec celles de l'ensemble de test.

D'après les tableaux nous pouvons comparer nos résultats avec ceux du MOTChallenge. Les valeurs du MOTA obtenues pour la séquence S2L1 de PETS 2009 se situent dans la moyenne basse par rapport aux autres approches, alors que pour la séquence *Sunny Day* de ETH dans la moyenne supérieure. La séquence S2L1 a été évaluée par plus d'approches différentes car elle appartient à un autre challenge de la communauté PETS. Chaque approche apporte une petite amélioration par rapport à la précédente.

Alors que la séquence *Sunny Day* est évaluée par un nombre plus restreint d'approches, qui ne sont pas les meilleures de l'état de l'art pour le suivi. L'ensemble du système proposé n'a pas pour objectif d'être compétitif ni de se positionner par rapport à la littérature, mais de montrer comment le même système peut se surpasser via l'usage de paramètres finement

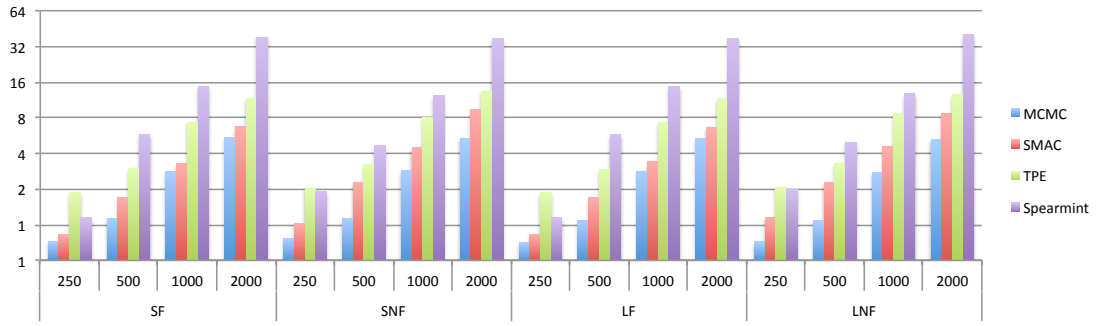


FIGURE 3.9 – Temps de calcul, en heures pour la séquence PETS S2L1.

choisis. Dans le tableau 3.3 nous observons une différence de plus de 2 pp entre le meilleur et le pire résultat. Cela peut sembler peu mais si nous examinons le classement du MOTChallenge, les scores entre différentes approches ne diffèrent que de quelques décimales. C'est pourquoi que même une petite amélioration sur quelques séquences peut mener au gain de 5 places dans le classement. En général les outils d'optimisation sont satisfaisants mais parmi ceux étudiés SMAC nous donne un paramètre plus juste. Si le but principal est d'améliorer la justesse nous recommandons son usage.

Nous analysons également l'impact de l'optimisation des paramètres par rapport à d'autres métriques du CLEAR-MOT ou à une fonctions objective combinant plusieurs métriques. Cependant, la métrique MOTA combine déjà les faux positifs, les faux négatifs et les changement d'identités. C'est pourquoi au regard de ces autres métriques nous arrivons aux mêmes conclusions qu'avec l'analyse du MOTA exclusivement.

Temps de calcul Le temps nécessaire pour aboutir à des performances stables par chacun des outils joue un rôle important pour sélectionner lequel utiliser. Nous observons sur la figure 3.9 que le temps de calcul est similaire pour MCMC et Spearmint que l'on soit dans la configuration F ou NF. En revanche SMAC et TPE présentent un comportement différent pour le temps de calcul nécessaire. Le coût CPU est plus grand lorsque le nombre de particules n'est pas fixé, le tableau 3.5 met en lumière ce phénomène. La raison de ce comportement est liée au besoin pour ces deux outils de construire un modèle pour 7 paramètres et non 6. Le temps de calcul augmente en fonction du nombre de paramètres à optimiser. C'est avec SMAC que l'ajout d'un paramètre mène à la plus grande augmentation du temps de calcul.

Valeurs initiales Le choix des valeurs initiales influence l'issue de la plupart des outils d'optimisation. Une bonne approche devrait être en mesure de gérer ça et de converger vers un optimum global sans rester bloqué dans un extrema local. Quand nous comparons le tableau 3.6 et les tableaux 3.1 et 3.2, nous observons que SMAC et MCMC présentent un comportement similaire en dépit du choix des paramètres initiaux. De plus, l'analyse de la convergence de SMAC montre que la seule différence est le nombre d'itération qui varie en fonction de la distance à

la valeur optimale. Cependant MCMC a des difficultés à converger, restant sur des minimums locaux lors de plusieurs itérations, ainsi qu'un problème de sur-apprentissage.

Influence de la taille de l'ensemble d'entraînement L'augmentation de la taille de l'ensemble d'entraînement augmente les performances. Nos évaluations aboutissent à cette même constatation. Nous pouvons noter qu'à partir de 10% de l'ensemble des données, l'incrément est faible. C'est parce que les 10% des séquences évaluées décrivent assez bien le mouvement des piétons dans la scène.

Le nombre de particules Le nombre de particules joue un rôle important pour ce type de traqueurs. Pour un nombre de particules fixé, les performances sont meilleures lorsque plus de particules sont utilisées au dépend du temps de calcul. En analysant les paramètres évalués par les outils sans avoir fixé le nombre de particules, nous remarquons que nombre de bons candidates maintiennent un nombre peu élevé de particules entre 10 et 50.

Accessibilité Dans l'ensemble, les outils permettent de définir un intervalle de recherche pour chacun des hyper-paramètres mais seul SMAC et notre implémentation de MCMC permettent d'initialiser leurs valeurs. A l'inverse de TPE et Spearmint qui utilisent la valeur de la borne inférieure de l'intervalle, ce qui simplifie leur configuration. SMAC présente le meilleur ratio entre le coût CPU et les performances, suivi de TPE. Au contraire de Spearmint qui incrémente le temps de calcul à chaque itération. C'est un problème car les résultats sont meilleurs plus il y a d'itérations. TPE a l'avantage de pouvoir être appelé comme une fonction. En comparaison, SMAC et Spearmint sont plus complexes, ils requièrent de créer des fichiers et dossiers sous un format spécifique. La documentation est cruciale lorsque l'on utilise un nouvel outil. De part cet aspect SMAC surpasse les autres avec des documentations officielles détaillées, des discussions actives sur des forums et d'un support technique pour les développeurs. MCMC dispose de larges bases théoriques dans la littérature avec de nombreuses adaptations et exemples. La documentation de TPE est moins exhaustive mais satisfaisante pour son usage. Au contraire de Spearmint dont la documentation est quasiment inexistante, principalement limitée à son installation et la lancement d'un exemple basique.

En outre, SMAC et Spearmint collectent des informations à chaque itération, qui sont disponibles pour que l'utilisateur puisse les observer et les analyser. SMAC est plus organisé, séparant les données spécifiques dans de nombreux fichiers faciles à comprendre. Parallèlement, Spearmint fournit des fichiers récapitulatifs de chaque itération et une base de données contenant les informations globales. TPE et MCMC ne fournissent pas des informations aussi exhaustives que les deux autres, mais des résultats simples tels que les paramètres et la valeur de la fonction de coût à chaque itération.

Synthèse À partir de ce qui précède, nous avons créé le tableau 3.9 qui résume notre expérience dans l'utilisation de ces outils. Nous considérons que SMAC offre le meilleur compromis en ce qui concerne les critères mentionnés sur le tableau 3.9. Il est le plus facile à utiliser malgré

le nombre de fichiers à configurer. La documentation est complète et détaillée et elle est soutenue par une communauté active. En outre, SMAC se révèle plus stable tant en ce qui concerne l'efficacité de la convergence que dans l'exploration, ce qui est important pour la répétabilité des résultats. Nous considérons TPE comme une bonne alternative, mais nous le classons en deuxième position. Il offre un large éventail de possibilités, mais la documentation limitée rend son utilisation difficile pour différents scénarios. Il a également un bon taux de convergence, dépassant même le SMAC dans certains cas. Spearmint donne de bonnes performances, selon nos résultats, mais il reste inférieur à SMAC et à TPE. De plus, la faible documentation et le temps de calcul sont des caractéristiques qui rendent son utilisation difficile. Cependant, le MCMC présente la plus mauvaise performance et nous le classons donc comme numéro quatre. Dans ce cas, le MCMC a du être implémenté. Les résultats sont satisfaisants et c'est le plus rapide de tous les outils, mais il est difficile de fixer le nombre correct d'itérations pour éviter le sur-apprentissage (*overfitting*).

Critère	MCMC	Spearmint	SMAC	TPE
Convergence	+	++	+++	+++
Stabilité	+	++	++	+++
Justesse	+	++	+++	++
Coût CPU	+++	+	+++	++
Conditions initiales	+	n/a	+++	n/a
Taille de l'entraînement	+	++	+++	+++
Accessibilité	+	+	+++	++
Rang	4	3	1	2

TABLE 3.9 – Synthèse des évaluations des différents outils d'optimisation.

3.5 Conclusion

Dans ce chapitre, nous avons présenté une étude comparative de quatre approches d'optimisation des hyper-paramètres pertinentes dans le contexte du MOT. Les outils sont examinés en fonctions de critères de performance, accessibilité, temps de calcul, entre autres. Nous avons montré comment le même peut fournir de meilleurs résultats en utilisant une meilleure combinaison de paramètres, et comment les trouver à l'aide d'outils spécialisés. Notre objectif est de présenter ces quatre méthodes d'optimisation, avec leurs outils respectifs, et de motiver leur utilisation. Nous avons mis en évidence les points forts et les faiblesses de chacun d'entre eux aussi détaillés que possible en tenant compte de nombreux critères. Les outils sélectionnés au moment de l'étude se sont avérés pertinents au regard de leur nombre de citations qui a au moins doublé, avec notamment un intérêt de la part de la communauté vision notamment pour l'optimisation des hyper-paramètres des réseaux de neurones qui sont aussi souvent des processus stochastiques. Depuis cette étude et suite au besoin grandissant d'optimisation des hyper-paramètres des approches deep-learning, d'autres bibliothèques d'optimisation ont vu le jour telle que Op-

tuna [Akiba 2019] en 2019. Il serait pertinent de l'évaluer et de la comparer avec les outils d'optimisation retenus ici.

La pertinence de cette étude nous a permis deux publications : IEEE AVSS 2017 et journal MVA 2019. Les références complètes se trouvent page vii.

CHAPITRE 4

Reconnaissance d'actions avec modèles bayésiens 3D

Sommaire

4.1 Introduction

La reconnaissance des activités humaines est une tâche importante dans le développement de nombreuses applications pratiques telles que le suivi de santé à domicile et la cobotique. Les activités menées à domicile ou dans un environnement industriel peuvent différer mais leurs actions sous-jacentes peuvent être similaires car elles induisent le déplacement d'objets, la manipulation d'objets, l'interaction avec et entre des objets... Comme nous l'avons vu dans le chapitre 2, ce sont les raisons pour lesquelles nous choisissons de raisonner à partir des données de positions des objets et de la trajectoire du squelette humain.

Il s'agit toujours d'une problématique complexe où les actions peuvent être mal interprétées en raison d'un environnement encombré, des occultations partielles du corps, de la présence de nombreux objets au voisinage immédiat de l'homme et des changements de point de vue. Dans

ce chapitre nous décidons de nous appuyer que sur un seul capteur RGB-D à bas coût tels que la Kinect ou la XTion.

Les approches basées sur les réseaux neuronaux convolutionnels (CNN) ont montré de bonnes performances dans le domaine de la détection d'objets et de l'estimation de la posture humaine sur des images en couleur. La fusion des percepts relatifs aux objets manipulés et à la posture humaine peut être alors exploitée pour interpréter des activités complexes, en particulier lorsque les objets sont en interaction avec l'environnement et que la trajectoire de la posture n'est pas hautement discriminante à elle seule. Ainsi, en nous appuyant sur les récentes avancées des CNN, nous proposons d'utiliser la posture humaine et la détection d'objets comme percepts de bas niveau pour notre approche bayésienne qui permet la reconnaissance d'actions à la volée. Ainsi dans la première partie de ce chapitre nous proposons une étude comparative de deux détecteurs de postures de l'état de l'art afin d'étayer notre choix.

Certaines actions ne diffèrent pas beaucoup dans la posture humaine impliquée mais plutôt dans la nature et l'état de l'objet durant l'exécution de l'action. Étant donné la classe inférée de chaque objet relative à sa nature, nous pouvons fournir a priori des connaissances sur la façon dont il est manipulé, comme décrit par la théorie de l'affordance [Gibson 1979]. Pour rappel, plusieurs types d'affordance peuvent être associées à une classe d'objet une pomme est un objet qui peut être déplacé, pelé, mangé mais ne peut pas être lu. Ces types d'affordance peuvent sous-entendre une configuration spécifique des objets vis-à-vis de leur environnement. Dans le cas de pelé il doit y avoir un éplucheur ou un couteau à proximité.

Cependant, la perception de la posture humaine est aussi pertinente pour effectuer la reconnaissance de l'action, car certaines actions ne peuvent être discriminées que par le mouvement associé aux membres du corps, ou du mouvement relatif de la posture par rapport aux objets. Par rapport aux autres membres du squelette. Par exemple, pour les actions de préhension il s'agit d'un mouvement typique de la main qui s'éloigne du corps pour atteindre un objet. En utilisant les informations de la posture, en particulier lorsque tous les objets sont statiques, nous pouvons trouver lequel est en interaction. De plus, les informations sur la posture humaine peuvent aider à réduire les erreurs dues aux mouvements erronés des objets lors du scintillement des détections dans l'image en profondeur.

Dans la deuxième partie de ce chapitre nous proposons d'intégrer la perception conjointe de l'homme avec : (1) sa posture exprimée au travers de la position de ses articulations et (2) les objets au travers de leurs nature et positions, à des modèles bayesiens pour la reconnaissance d'actions. Ces détections sont collectées via des détecteurs de la littérature : OpenPose [Cao 2017] et SSD (*Single Shot MultiBox Detector*) [Liu 2016a]. Les modèles d'actions sont crées de part notre connaissance experte et ne requièrent pas de larges jeux de données. De plus ils peuvent être réutilisables dans différents contextes. Nous allons également exploiter l'outil SMAC présenté dans le chapitre précédent pour régler certains paramètres.

Nous proposons dans la section 4.2 une évaluation de deux détecteurs de la littérature sur des jeux de données acquis avec un système de capture de mouvement, puis dans la section 4.3 nous présentons notre approche pour la reconnaissance d'actions au moyen du squelette humain et des objets en interaction, la section 4.4 présente les résultats obtenus sur deux jeux de données

(a) Berkeley MHAD

(b) Human 3.6M (H3M)

FIGURE 4.1 – Bases de données publiques avec capture de mouvement.

publics de la littérature et enfin la conclusion en section 4.5.

4.2 Évaluations de détecteurs de posture

Rappelons que l'on cherche à reconnaître les actions non pas directement à partir de caractéristiques issues données brutes c.à.d du flux vidéos mais grâce à des percepts de plus haut niveau. Nous retenons l'évolution de la posture et de la trajectoire des différents objets à travers le temps. Plusieurs détecteurs de posture existent dans la littérature, nous commençons logiquement par une étude comparative de leurs performances.

4.2.1 Détecteurs de posture et jeux de données

Si en 2020 l'utilisation de OpenPose [Cao 2017] s'est généralisée auprès de la communauté vision ce n'était pas le cas en 2017. Il n'existe que peu d'études comparatives en dehors des jeux de données 2D [Sarafianos 2016] et évaluant OpenPose. Au moment de cette étude, cette approche récente publiée dans CVPR 2017 n'avait été citée que 50 fois contre + 2400 fois en 2020. D'où l'intérêt de mener une étude comparative de ses performances par rapport aux détecteurs de l'état de l'art en 2017. De plus si de nombreux jeux de données permettent d'évaluer les détecteurs de mouvements en 2D, elles ne fournissent pas de positions 3D réelles. Il existe donc peu d'études comparatives en 3D. Nous proposons une évaluation des ces détecteurs par rapport à une vérité terrain 2D et 3D acquise au moyen d'un système de capture de mouvement (*Motion Capture*). Nous avons également considéré DeeperCut [Insafutdinov 2016] qui détenait la seconde position dans l'état de l'art sur un jeu de données de référence pour la reconnaissance de posture : MPIIMulti-Person [Andriluka 2014]. Il existe d'autres détecteurs [Iqbal 2016, Pishchulin 2016] mais nous ne les avons pas évalués vis-à-vis des évaluations antérieurs menées par d'autres articles. De plus nous avons privilégié les approches dont le code était disponible.

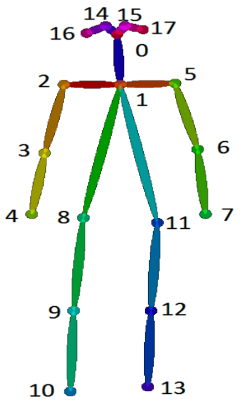


FIGURE 4.2 – Estimation de la posture par OpenPose avec les 18 articulations considérées.

OpenPose et DeeperCut sont deux approches qui détectent d'abord les différentes parties du corps humain puis cherchent à les associer pour reconstruire la posture. Elles ne reposent donc pas sur une première détection des personnes. Ces détecteurs fonctionnent qu'il y ait une ou plusieurs personnes dans l'image, il reste à résoudre le problème d'association de ces articulations sans savoir le nombre de personne dans l'image. Afin de résoudre le problème d'association des différentes parties du corps détectées DeeperCut propose une optimisation incrémentale mais cela prend encore quelques minutes par image. OpenPose quand à lui, pour l'entraînement, ajoute à la vérité terrain des positions des articulations des champs d'affinité par partie aux différents membres du corps. Ce champs d'affinité est un vecteur 2D qui encode la direction d'un côté du membre vers l'autre. Cette étape d'entraînement joint de la reconnaissance des articulations permet d'accélérer l'ensemble du processus et ce indépendamment du nombre de personnes présentes dans l'image. Au final OpenPose retourne les coordonnées image des positions prises par les articulations du squelette avec un indice de confiance. Les articulations considérées apparaissent sur la figure 4.2.

La comparaison de OpenPose à DeeperCut est réalisée sur des jeux de données 2D, tel que MPIIMulti-Person. Nous proposons une étude sur des jeux de données enregistrés au moyen de capteurs RGB-D et d'un système de capture de mouvement afin d'obtenir une vérité terrain des positions des articulations dans un espace tri-dimensionnel.

Nous proposons ici une évaluation de leur performance sur deux jeux de données avec Mo-Cap introduits dans le chapitre 2 : Human 3.6M [Ionescu 2014] et Berkeley MHAD [Ofli 2013]. Ces deux jeux de données se prêtent bien à cette évaluation car ils ne contiennent pas d'objets et l'environnement se cantonne au système d'acquisition, avec l'acteur au centre et un arrière plan qui ne contient pas d'informations relatives à l'action qui se joue, comme le montre la figure4.1.

4.2.2 Évaluations de OpenPose et de DeeperCut

Sur les deux jeux de données nous obtenons les prédictions des deux détecteurs : OpenPose et DeeperCut. Les positions 2D des articulations détectées sont ensuite projetées en 3D grâce à

FIGURE 4.3 – Erreur moyenne sur l'ensemble des articulations, en pixels, pour les acteurs 5, 8, 9, et 11 entre la vérité terrain et les prédictions d'OpenPose (en bleu) et les prédictions de DeeperCut (en rouge) sur le jeu de données Human 3.6M.

FIGURE 4.4 – Erreur moyenne pour chacune des articulations, en pixels, entre la vérité terrain et les prédictions d'OpenPose (en bleu) et les prédictions de DeeperCut (en rouge) sur le jeu de données Human 3.6M.

l'étalonnage préalable des caméras. Les positions 3D ainsi obtenues sont comparées aux positions 3D des articulations correspondantes obtenues grâce au système de capture de mouvement (*Motion Capture*). Ceci nous permet de calculer la distance en millimètres entre la prédiction et la vérité terrain. Nous proposons également une comparaison en 2D dans le plan image en exprimant les distances en pixels. Nous évaluons les détecteurs en fonction des différents acteurs mais également en fonction des articulations et des séquences d'actions.

Les résultats présentés sur la figure 4.3 concernent les acteurs 5,8,9 et 11 du jeu de données Human 3.6M [Ionescu 2014]. La performance moyenne d'OpenPose est moins bonne que celle de DeeperCut. Les distances étant exprimées en pixels, notons qu'il n'y a en moyenne qu'un écart de 1.7 pixels entre les deux approches. Ces différences de performances entre les acteurs sont considérées comme négligeables, nous allons donc analyser les performances des détecteurs par rapport aux articulations individuelles.

Analysons maintenant les performances pour chacune des articulations, les résultats sont présentés sur la figure 4.4. Nous observons que généralement les performances de OpenPose et DeeperCut sont équivalentes pour l'ensemble des articulations sauf pour le coude droit et le genou droit. C'est en réalité en analysant les résultats par rapport aux différentes séquences d'actions que nous nous apercevons que OpenPose estime le plus mal leurs positions lorsqu'il y a de nombreux mouvements au sol comme avec l'action *Purchases* et *Sitting Down*. Lors de ce type d'actions ces articulations se retrouvent le plus souvent en situation d'auto-occultations du à l'angle de la caméra par rapport à l'acteur.

FIGURE 4.5 – Erreur moyenne sur l'ensemble des articulations, en millimètres, pour chacun des acteurs (1-5) entre la vérité terrain et les prédictions de OpenPose (en bleu) et les prédictions de DeeperCut (en rouge) sur le jeu de données Berkeley MHAD.

Nous présentons ci-après les résultats sur cinq acteurs du jeu de données Berkeley MHAD sur la figure 4.5. Il y a en moyenne un écart de 13.7 cm entre la prédiction de DeeperCut et la vérité terrain, et un écart de 14.2 cm pour les prédictions de OpenPose. Focalisons nous alors sur les performances pour chacune des articulations. La différence de performance entre les deux détecteurs est inférieure à 1 cm (0.5 cm) lorsque l'on compare les performances par rapport aux différents acteurs.

La figure 4.6 montre que les détecteurs sont moins précis pour les articulations du côté droit.

FIGURE 4.6 – Erreur moyenne pour chacune des articulations, en millimètres, entre la vérité terrain et les prédictions de OpenPose (en bleu) et les prédictions de DeeperCut (en rouge) sur le jeu de données Berkeley MHAD.

Ceci s'explique par l'angle de la caméra qui induit des situations d'auto-occultations relatif au côté droit des acteurs, voir figure 4.1a. De plus il y a souvent de larges erreurs sur le poignet gauche (poignet G). Le tableau 4.1 donne les performances des deux détecteurs en fonction des différentes actions du jeu de données Berkeley MHAD. En moyenne les performances sont similaires avec donc un léger avantage pour DeeperCut mais cela ne concerne pas l'ensemble des actions, OpenPose est meilleur sur certaines actions. Après une analyse plus fine en regardant également les séquences d'actions, il s'agit surtout de l'action 2 qui souffre de larges écarts avec la vérité terrain. C'est parce que lors de cette action les acteurs sautent et leur main gauche sort souvent de du champ de la caméra comme illustré sur la figure 4.1a.

TABLE 4.1 – Moyenne et écart-type des erreur en millimètres sur le jeu de données Berkeley MHAD pour chacune des différentes actions .

Détecteur	Action	Distance	σ	Détecteur	Action	Distance	σ
	1	102	53		1	97	43
	2	190	191		2	200	209
	3	248	268		3	297	336
	4	149	115		4	171	142
	5	185	185		5	174	167
DeeperCut	6	94	48	OpenPose	6	98	55
	7	112	66		7	108	63
	8	115	44		8	116	45
	9	107	80		9	99	41
	10	103	70		10	92	36
	11	404	77		11	91	54
Moyenne		**137**	**109**	Moyenne		**140**	**108**

L'ensemble des évaluations menées jusqu'alors montrent des performance équivalentes pour les deux détecteurs : DeeperCut et OpenPose. Cependant le temps de calcul de ces deux approches est très différent. Les évaluations sont menées sur un ordinateur équipé d'une carte graphique NVIDIA 1080Ti. Il en ressort que OpenPose est bien plus rapide avec des prédictions à 13 trames par seconde (fps) alors que DeeperCut est 43 fois plus lent avec seulement 0.3 fps.

Pour la reconnaissance de la posture humaine au travers de la détections des différentes articulations du squelette s associé nous utiliserons OpenPose car il est plus rapide pour des performances comparables. De plus de nombreuses fonctionnalités sont développées et utilisables si besoin avec par exemple une détection fine des mains.

4.3 Modèles bayésiens pour la reconnaissance d'actions

Dans cette section nous définissons les modèles probabilistes qui sont au cœur de notre processus de reconnaissance d'action, puis nous détaillons leur implémentation. L'approche décrite si après est abrégée ANBM pour A New Bayesian Model.

4.3.1 Définition des modèles probabilistes

Dans ce chapitre nous nous intéressons aux actions réalisées par une personne seule avec plusieurs objets à sa disposition. La nature des objets et les interactions possibles entre ces objets et entre la personne et les objets sont connus *a priori*. La reconnaissance d'actions s'opère à partir de l'observation conjointe de la posture humaine et des objets présents dans la scène. Nous décrivons la posture humaine au moyen d'un squelette composé d'un ensemble de points d'articulation et de leurs coordonnées 3D. Certaines paires d'articulations représentent un membre du corps, par exemple (le coude, le poignet) pour l'avant-bras.

Nous considérons qu'il y a deux types de mouvements corporels : pré et post-saisie d'un objet, avec d'une part des mouvements d'approche et de l'autre des mouvements liés à la manipulation. Les premières ont lieu avant que la main ne capture un objet et les autres ont lieu lors du contact entre la main et un objet. Dans les deux cas, l'ensemble des objets environnants et la posture humaine influencent les probabilités des différentes classes d'actions. Les affordances des objets sont prises en compte dans les modèles d'interactions homme-objets ainsi que dans les modèles d'interactions objet-objet. Une fois que l'objet est saisi, il peut continuer d'interagir soit avec une partie du corps comme la bouche pour manger un fruit, l'oreille pour téléphoner, ou alors avec un autre objet comme lors de l'action verser du lait dans un bol, placer un livre sur une table. Il peut également ne pas y avoir d'autres interactions comme pour les actions de déplacement d'un objet, d'utilisation d'un smartphone, d'allumage d'un interrupteur... Notre approche est construite pour prendre en considération chacun de ces différents types d'actions dans un processus bayésien unifié à partir d'observations de la configuration spatiale des objets et de la posture humaine comme illustré sur la figure 4.7.

Comme nous l'avons vu dans l'état de l'art au ch apitre 2, nous choisissons d'abord d'explorer une approche dite paramétrique. En effet, nous choisissons d'utiliser les deux modalités de couleur et de profondeur. Les jeux de données RGB-D que nous avons choisis contiennent une quantité de données plus faible. C'est une première raison pour laquelle nous privilégions les approches paramétriques par rapport aux approches d'apprentissage profond qui sont réputées pour nécessiter de nombreuses données annotées. Nous souhaitons aussi développer une approche dont les résultats sont plus facilement interprétables et pouvoir tirer parti de contraintes liées aux affordances ou à la sémantique des verbes d'actions.

À chaque action a nous y associons un modèle. Soit $\mathrm{A} = \{a^1, a^2, ..., a^n\}$ l'ensemble des actions. Soit $O_t = \{s_t, \Omega_t\}$ une représentation de l'ensemble de l'observation conjointe du la posture humaine s_t et de l'ensemble des objets détectés aux abords de la personne : $\Omega_t = \{\omega^1, \omega^2, ..., \omega^{Card(\Omega)}\}$ avec $Card(\Omega)$ étant le nombre d'objets présents dans la scène. Nous modélisons la probabilité *a posteriori* des actions sachant les observations comme suit :

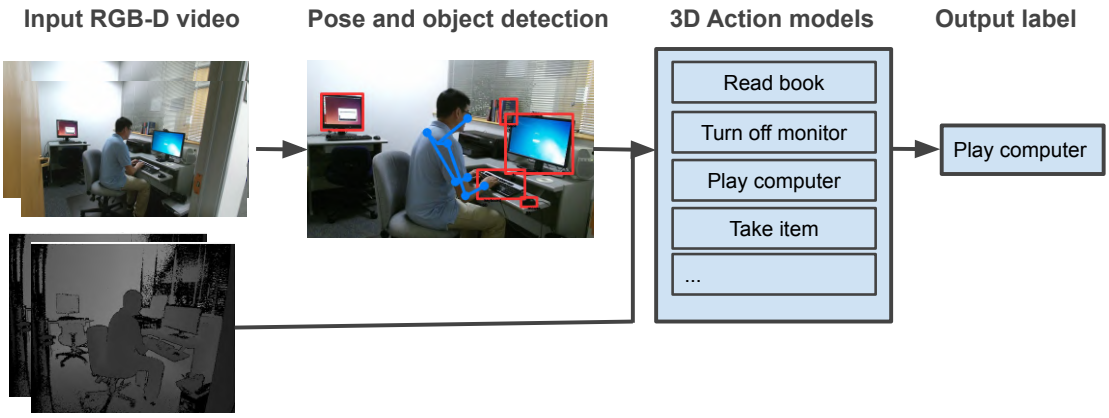


FIGURE 4.7 – Schéma-bloc de notre approche avec en entrées le flux RVB-D, la détection 2D de la posture et des objets. Ce sont ces données qui permettent le calcul du modèle d'action le plus probable et d'aboutir à la détection d'une action.

$$p(a_{0:T}|O_{0:T}) \propto \prod_{t=0}^{T} p(O_t|a_t) \prod_{t=1}^{T} p(a_t|a_{t-1}). \tag{4.1}$$

Dans l'équation (4.1), $p(O_t|a_t)$ est la vraisemblance de l'observation O_t sachant l'action a_t. Le terme $p(a_t|a_{t-1})$ désigne les probabilités de transitions entre deux actions. Ainsi notre approche a pour objectif de reconnaître à la volée les actions, sans utiliser une pré-segmentation des actions comme pré-traitement. La classification s'effectue au travers de l'agrégation d'information sur T trames. L'inférence se fait en temps presque réel car T est généralement compris entre 5 et 15 trames consécutives. L'intérêt de la fenêtre temporelle est double, elle permet l'agrégation d'informations et d'effectuer une localisation et reconnaissance d'actions.

L'idée d'ajouter une matrice de transition est motivée par l'observation suivante. Comme nous pouvons observer sur le tableau 4.2 décrivant les transitions possibles entre l'action courante (sur les colonnes) et l'action suivante (sur les lignes) sur le jeu de données CAD-120, de nombreuses transitions n'existent pas. En effet, l'enchaînement des actions est conditionné par une certaines logique lors de l'exécution d'activités. Par exemple un objet qui n'a pas été saisi ne peut pas être déplacé. L'action atteindre (*reach*) est suivie dans 69,2 pourcent du temps par l'action déplacer *move*. De nombreuses actions sont souvent suivies de l'action déplacer (atteindre, verser, manger, boire, nettoyer). Alors que les actions d'ouvrir, de placer et null sont plus fréquemment suivie par l'action atteindre. La vraisemblance d'observation dans l'équation (4.1) se décompose comme suit :

$$p(O_t|a_t) \propto p(s_t|a_t, \Omega_t)p(\Omega_t|a_t). \tag{4.2}$$

Ici dans l'équation (4.2), $p(s_t|a_t, \Omega_t)$ modélise la probabilité que l'observation du squelette s_t, soit une représentation de l'estimation du modèle de squelette associé à l'action a_t en interaction avec un des objets de l'ensemble Ω. La représentation du squelette est dans notre cas réduite

aux positions 3D du haut du corps avec les mains et la tête. Le squelette est représenté par un vecteur de coordonnées cartésiennes de dimension 9 dans le repère du monde comme montré par l'équation (4.3) :

$$s = (s_{\text{l.hand}},\ s_{\text{head}},\ s_{\text{r.hand}}). \tag{4.3}$$

L'équation (4.2) est le produit de deux vraisemblances, la première concerne l'observation de la posture et peut s'exprimer ainsi :

$$p(s_t|a_t, \Omega_t) \propto \mathcal{N}(d_p(s, s_n); \mu_1, \sigma_1)\mathcal{N}(d_o(s, \Omega); \mu_2, \sigma_2), \tag{4.4}$$

où s_n est le modèle de squelette associé à l'action a_t. Par exemple si $a_t =$ manger, l'une des deux mains se doit d'être proche de la tête. Dans le cas où $a_t =$ ouvrir, la main est éloignée de la tête et s'en rapproche car il s'agit d'un mouvement de flexion du bras, à l'inverse pour l'action $a_t =$ fermer qui est lui représenté par un mouvement d'extension du bras. Pour des raisons de clarté le terme t est omis par la suite. Dans cette équation, d_p est la distance 3D entre le modèle du squelette et le squelette estimé.

La deuxième distribution normale de l'équation 4.4 représente la probabilité qu'un objet dans la main soit en interaction avec d'autres objets dans la scène. La distance 3D entre les mains du squelette et les objets est décrite par d_o. Les écarts-types σ_1 et σ_2 de cette équation sont des paramètres à régler pour chaque action. L'affordance des objets et les interactions entre objets sont modélisées par la position des mains et les distances relatives inter-objets.

La deuxième vraisemblance de l'équation (4.2), $p(\Omega_t|a_t)$ représente la configuration spatiale de l'ensemble des objets de la scène et est modélisée comme suit :

$$p(\Omega_t|a_t) = \prod_{k=1}^{\text{Card}(\Omega_t)} p(\omega_t^k|a_t). \tag{4.5}$$

Dans l'équation (4.5), la vraisemblance est également exprimée comme une distribution normale. Les différents modèles sont définis grâce à l'analyse des actions par un utilisateur expert. Par exemple pour $p(\Omega_t|\text{déplacer})$, tous les objets peuvent être déplacés sauf le micro-ondes et la table. L'objet ω_{k2} supposé en interaction doit avoir une certaine distance par rapport à la table.

Les autres objets de la scène doit être immobiles (ou une distance à la table qui est nulle) caractérisés par : $p(\Omega_{\setminus\omega_{k2}}|\omega^{k2}) = \mathcal{N}(|l_\omega^{k2} - l_{\text{table}}|, 0, \sigma)$.

Dans le cas de l'action *placer*, $p(\Omega_t|\text{placer})$. Ici, l'ensemble des objets ne peuvent pas être placés ni déplacés. C'est le cas du micro-onde et de la table. Nous pouvons atteindre le micro-ondes mais il n'est pas possible de déplacer le micro-onde dans ce jeu de données. L'ensemble $\Omega_{\setminus\omega_{k2}}$ contient l'objet sur lequel va être placé celui qui est en interaction dans la main. Cela peut être l'objet table, ou d'autres objets. Pour être dans le cas de l'action *placer* l'objet supposé en interaction doit se rapprocher d'un autre objet de la scène et la distance entre eux diminue.

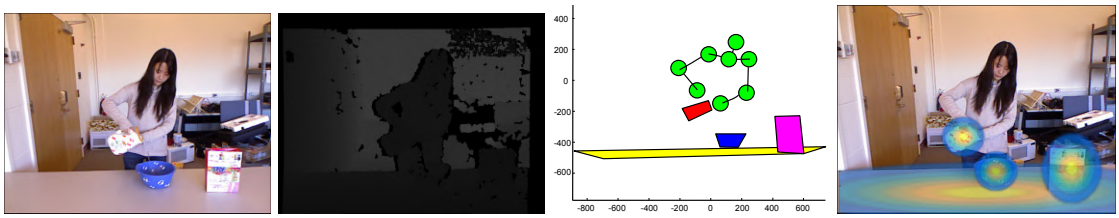
Le calcul de $p(\omega^{k2}|a^j = \text{eat})$ représente un cas où l'utilisateur expert ajoute une information d'a priori sémantique. Il est issu de définitions des actions qui sont possibles de réaliser avec des objets. Puisqu'il est possible de manger une pomme mais pas de boire une pomme il vient :

TABLE 4.2 – Statistiques sur le séquencement entre deux actions dans le jeu de données CAD-120. L'action courante est sur la colonne et l'action suivante sur les lignes.

	atteindre	déplacer	verser	manger	boire	ouvrir	placer	fermer	nettoyer	null
atteindre	1	2.5	0	30.7	0	**94.2**	**64.5**	0	0	**61.3**
déplacer	**69.2**	1.7	**100**	**69.2**	**100**	3.8	11	3.6	**100**	16.7
verser	0	9	0	0	0	0	0	0	0	0
manger	0	11	0	0	0	0	0	0	0	0
boire	0	9	0	0	0	0	0	0	0	0
ouvrir	17.9	0.3	0	0	0	1.9	0	0	0	0
placer	0	**52.5**	0	0	0	0	0	0	0	0
fermer	11.8	0	0	0	0	0	1.2	0	0	0.7
nettoyer	0	3.4	0	0	0	0	0	0	0	0
null	0	10.5	0	0	0	0	23.3	**96.4**	0	21.3

$p(\text{pomme}|\text{manger}) = 1$ et $p(\text{verre}|\text{manger}) = 0$. Nous proposons de calculer la distance entre la position de l'objet l_ω^{k2} et la position de la tête. l_{head}.

La figure 4.8 présente les trames RGB-D d'entrée, la configuration spatiale des objets en 3D ainsi que leurs distributions de probabilité gaussienne. Nous pouvons voir que lors de l'action *verser* les objets statiques sont en interaction avec la table et que la bouteille de lait est en interaction avec le bol tout en étant en interaction avec la main.



(a) (b) (c) (d)

FIGURE 4.8 – Exemple de trame issu du jeu de données CAD-120 est extraite de l'action *verser*. De gauche à droite : image RGB, image de profondeur D, modélisation des observations en 3D, gaussiennes associées.

4.3.2 Implémentation

Comme mentionné précédemment, notre formulation requiert un squelette s qui est une représentation de la posture humaine et les objets ω observés. Ils sont tous deux détectés à partir des images 2D au moyen de deux réseaux deep-learning de la littérature. Les détections effectuées sur les images 2D sont ensuite projetées vers le repère monde 3D de la caméra Kinect à partir de ses données de calibration.

La posture humaine 2D est inférée par OpenPose [Cao 2017], une bibliothèque open-source avec plus de 900 citations en 2018 et en 2020 plus de 2400, il propose un bon compromis entre les performances sur les jeux de données avec capture de mouvement et le temps d'exécution comme évoqué en section 4.2.1. Les auteurs ont également fait paraître les modèles pré-entraînés sur le jeu de données MSCOCO [Lin 2014a] keypoints. Cette bibliothèque prend en entrée une image RGB afin de détecter la posture des personnes présente c'est à dire la localisation image des 25 articulations du squelette et leurs scores de confiance associés. Cette approche *bottom-up* ne repose pas sur la détection de la personne en premier, mais plutôt des différents membres du corps et de leur association. Il a été montré qu'il offre de bonnes performances dans les situations d'auto-occultation et qu'il infère des prédictions en presque temps réel suivant les configurations.

Les objets de la scène sont détectés par SSD [Liu 2016b] pré-entraîné sur le jeu de données MSCOCO [Lin 2014a] comprenant 80 catégories d'objets. Les catégories incluent des objets de la vie courante tels que : un bol, une bouteille, un micro-onde, une télévision, un bureau, un smartphone. C'est un détecteur reconnu de la littérature avec plus de 3400 citations en 2018 et 9510 citations en 2020.

Comme énoncé dans la section 4.3, nous devons définir des valeurs pour la matrice de transition entre actions et pour chaque densité de probabilité gaussienne. Lors des interactions objet-objet, cela dépend de la distance entre deux paires d'objets, par exemple lorsque l'on verse du lait dans un bol. Dans le modèle d'affordance, cela peut dépendre de la distance entre la tête et l'objet par exemple lors de l'action boire de l'eau ou dépendre également de la vélocité. La matrice de transition entre actions a aussi besoin d'être définie, elle est de taille $n \times n$, n étant le nombre d'actions.

Le paramétrage manuel d'un grand nombre de paramètres est difficile du à la dimensionnalité de l'espace de configuration, les méthodes de recherche par exhaustives ne sont pas adaptées. Nous privilégions donc une méthode d'optimisation des hyper-paramètres présenté dans le chapitre 2 : SMAC [Hutter 2011a]. Afin de trouver des valeurs d'un jeu de paramètres, cela nécessite une mesure de performance de la fonction à optimiser. Pour rappel, SMAC optimise les paramètres d'une fonction-objectif par rapport à une fonction de coût. Dans notre cas, nous voulons trouver l'ensemble des paramètres qui mènent au meilleur résultat en termes de classification d'actions. Nous privilégions donc le F_1-score : f_{a^i} pour chaque action $a \in \mathrm{A}$, avec n étant le nombre d'actions à considérer.

Ainsi nous définissions la fonction de coût C par l'équation (4.6) :

$$C = n - \sum_{i=1}^{n} f_{a^i}. \tag{4.6}$$

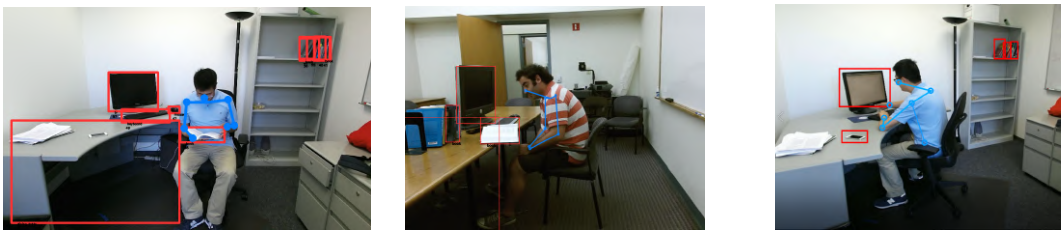
Comme mentionné dans le chapitre 3, SMAC construit un modèle de la fonction-objectif à chaque itération. C'est ce qui lui permet de trouver un ensemble optimal de paramètres avec peu d'évaluations. SMAC prend en compte l'issue des précédentes évaluations afin d'éviter de prendre du temps à évaluer des paramètres infructueux, contrairement à des méthodes de recherche exhaustives. Nous comparons les actions détectées à la vérité terrain pour calculer les

F_1-scores et donc la fonction de coût C.

4.4 Évaluations

Cette section précise les deux jeux de données publics et les métriques utilisés pour évaluer notre approche. Puis nous présentons et discutons des résultats obtenus par rapport à la littérature.

4.4.1 Jeux de données et métriques



(a) Vue de face (b) Vue de côté (c) Vue de trois-quart dos

FIGURE 4.9 – Variation des points de vue de la caméra par rapport à la position de l'acteur sur le jeu de données Watch-n-Patch.

Nous évaluons notre approche sur deux jeux de données de la littérature : CAD-120 [Koppula 2013b] et Watch-n-Patch [Wu 2015b] présentés dans le chapitre 2. Ainsi nous pouvons nous comparer avec les autres approches de l'état de l'art. Dans la littérature les approches s'évaluent et se comparent avec deux métriques principales : F_1-score et la justesse définie comme le micro-précision et rappel. Pour rappel le F_1-score se définit comme suit :

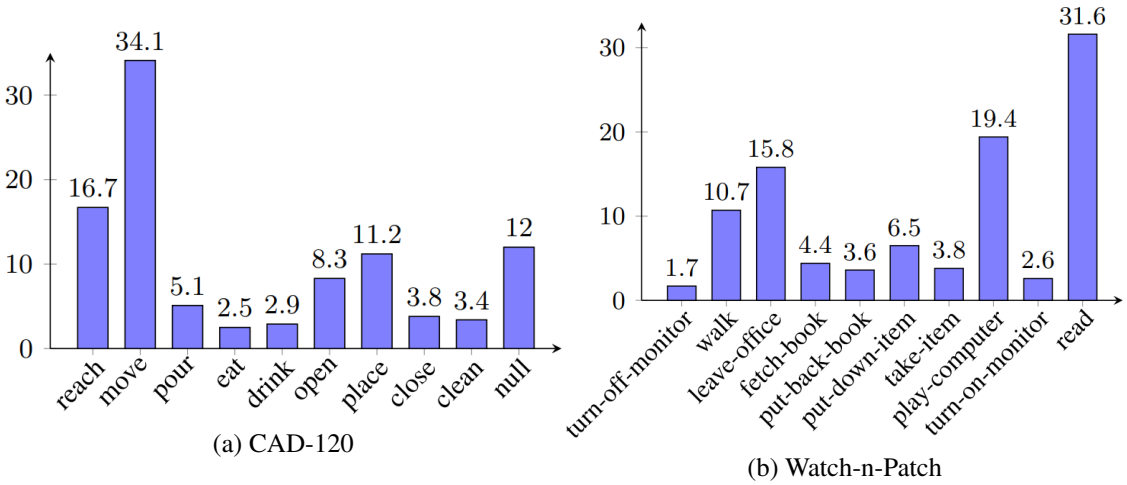
$$\text{F}_1\text{-score} = 2 \times \frac{\text{P} \times \text{R}}{\text{P} + \text{R}}, \tag{4.7}$$

P étant la précision et R le rappel de notre système de classification. Notre approche ne prédit qu'une seule classe d'action par trame, dans ce cas la justesse représente le ratio de trames correctement prédites, aussi appelée justesse-par-trame (*frame-wise accuracy*).

Puis les approches en comparaison fournissent souvent des détails à propos de la macro précision et du macro rappel. La différence de calcul entre le micro et la macro dépend de la prise en compte du fait que les classes du jeu de données soient balancées ou non. Les métriques sont calculées avec ou sans prise en compte de la segmentation temporelle des actions dépendant de la nature des approches.

Pour une comparaison juste des approches sur CAD-120 nous suivons le protocole recommandé par les auteurs du jeu de données : une validation croisée avec 4 plis. L'entraînement se fait sur 3 acteurs et un acteur est laissé de côté pour la phase de test, à l'instar des ap-

proches [Hu 2014, Koppula 2013b, Qi 2017]. Nous utilisons l'outil SMAC pour paramétrer les hyper-paramètres sur chacun des plis d'entraînement.



(a) CAD-120

(b) Watch-n-Patch

FIGURE 4.10 – Diagrammes en barre représentant la répartition en % des différentes actions au sein des deux jeux de données : CAD-120 à gauche et Watch-n-Patch à droite.

4.4.2 Résultats et discussion

Ci-après nous comparons notre approche pour la reconnaissance d'actions à l'état de l'art. Pour rappel, celle-ci gère la reconnaissance d'actions en ligne et leurs transitions. Nous nous évaluons de deux manières différentes avec et sans pré-segmentation des actions afin de se comparer au plus grand nombre d'approches de la littérature.

Puisque notre approche prédit une action par trame et non sur l'ensemble d'une séquence, nous l'adaptons légèrement comme suit. Puisque les bornes temporelles des séquences sont connues, nous utilisons un mécanisme de vote pondéré pour associer ce segment à une seule classe. La précision moyenne, le rappel et le F_1-score sont calculés à partir de la distribution des classes d'actions sur la figure 4.10a et des matrices de confusion trouvées dans les publications originales quand elles sont publiées.

Les résultats avec la pré-segmentation des actions effectuée à partir de la vérité terrain sur CAD-120 [Koppula 2013b] sont résumés sur le tableau 4.3. Les approches [Qi 2018a, Jain 2016b] reposent sur l'entraînement des réseaux de neurones pour la prédiction. Alors que [Hu 2014] repose sur le calcul de maximum de vraisemblance sur des graphes et [Qi 2017] met en œuvre des grammaires stochastiques. Ces approches ont été présentées dans le chapitre 2. Nous montrons des résultats similaires à l'approche S-RNN [Jain 2016b] avec une différence de 0,01 dans le F_1-score, sans avoir besoin de grands jeux de données annotés.

En comparaison avec HELK [Hu 2014], nous obtenons des résultats similaires en moyenne mais ils montrent de meilleures performances dans la détection des actions atteindre et déplacer. Cependant ils montrent des performances moindres en matière de précision pour les actions

TABLE 4.3 – Précision (P), Rappel (R) et F_1-score (F1) moyennés sur toutes les classes du jeu de données CAD-120 [Koppula 2013b] en utilisant la pré-segmentation des actions fournie par la vérité terrain.

Approche	P	R	F1
GPNN [Qi 2018a]	0,88	0,86	0,87
HELK [Hu 2014]	0,83	0,82	0,82
S-RNN [Jain 2016b]	N/A	N/A	0,83
QHWZ [Qi 2017]	0,71	0,68	0,69
ANBM	**0,84**	**0,80**	**0,82**

manger et nettoyer, inférieures à 0,3, qui sont parmi les classes les moins représentées dans le jeu de données comme montré par le diagramme de la figure 4.10a. Ces deux classes représentent respectivement 2,5 et 3,4 % du jeu de données. Alors que nous maintenons de bonnes performances sur ces deux actions comme le montre la suite des évaluations.

Maintenant nous considérons une configuration plus exigeante où la vérité terrain de la segmentation des actions n'est pas disponible. Dans ce cas nous calculons l'action la plus probable à chaque trame en tenons compte des $T = 10$ trames précédentes et les résultats sont montrés par le tableau 4.4. Les macro-précision et macro-rappel sont issus des matrices de confusion des publications originales quand elles sont disponibles. La justesse est calculée comme le micro précision et rappel en prenant en compte la répartition des actions dans le jeu de données présentées sur la figure 4.10a. Dans CAD-120, la représentation des actions est très déséquilibrée : par exemple les actions atteindre et déplacer représentent plus de la moitié de l'ensemble des actions.

C'est pourquoi nous présentons la précision et le rappel pour chacune des actions pour une analyse plus fine des performances en dépit du déséquilibre des classes. En effet d'avoir de très bonnes performances sur l'action nettoyer influence peu la justesse globale d'une approche car elle n'est présente qu'à 3,4 %. Nous montrons une amélioration générale d'au moins 4 points de pourcentage (pp) en justesse. Cette amélioration est d'autant plus significative vis-à-vis des actions placer et nettoyer avec + 16 pp et + 30 pp dans le F_1-score. Nous avons enrichi les détections originales données par le dataset par les détections de SSD avec la détection de la table. Puisque les objets sont souvent placés sur la table, de connaître sa position aide à créée un modèle d'action plus précis associé à l'action placer.

Dans la matrice de confusion sur la figure 4.11a, nous observons une forte diagonale et notons que l'une des sources majeures d'erreur avec un fort taux de faux positifs est l'action déplacer (*move*). L'action du déplacement d'un objet est l'action la plus fréquente car elle précède systématiquement la plupart des actions : verser, manger, boire, place et nettoyer. Le taux de faux positifs est due aux transitions entre l'action déplacer et l'action suivante qui est moins discriminante.

TABLE 4.4 – Macro précision (P), macro rappel (R), et justesse pour chacune des différentes actions sur le jeu de données CAD-120 [Koppula 2013b] sans pré-segmentation effectuée au moyen de la vérité terrain.

Approche	KGS [Koppula 2013b]		KS [Koppula 2013a]		**Notre approche ANBM**	
Action	P	R	P	R	P	R
reach	N/A	N/A	0,63	0,65	0,67	0,81
move	N/A	N/A	0,30	0,86	0,47	0,68
pour	N/A	N/A	0,93	0,59	0,8	0,73
eat&drink	N/A	N/A	0,92	0,52	0,90	0,79
open&close	N/A	N/A	0,84	0,63	0,84	0,75
place	N/A	N/A	0,66	0,61	0,84	0,76
clean	N/A	N/A	0,46	0,58	0,88	0,83
Moyenne	0,71	0,62	0,75	0,63	0,80	0,76
Justesse		0,68		0,70		**0,74**

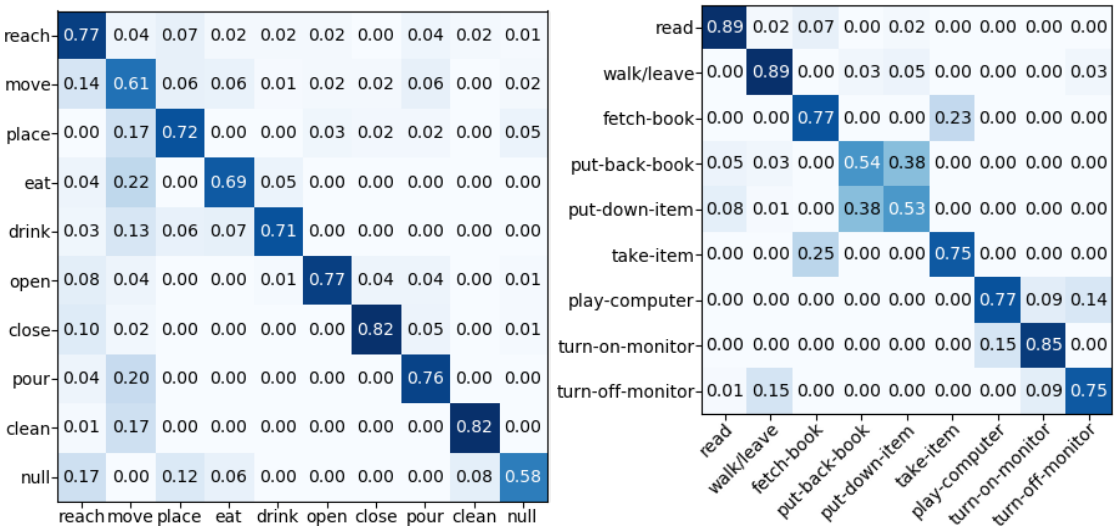
TABLE 4.5 – Comparaison avec les approches de l'état de l'art en termes de justesse en tant que pourcentage des trames dont les actions ont été correctement reconnues sans pré-segmentation sur le jeu de données Watch-n-Patch, environnement *office*.

Approche	Justesse
CaTM [Wu 2015b]	38,5
WBTM [Wu 2016]	41,2
PoT [Ryoo 2015]	49,93
KMHIS [Kataoka 2016]	59,75
Notre approche	**76,1**

Nous comparons nos résultats à la littérature qui s'évalue sur le jeu de données Watch-n-Patch. Ce qui démontre l'adaptabilité de la méthode que nous proposons car les deux jeux de données sont assez différents quant aux points de vue et des actions à reconnaître.

Le tableau 4.5 décrit le ratio des trames correctement annotées (justesse) comme dans [Wu 2015b, Wu 2016]. Ici encore, nous observons une forte amélioration par rapport aux approches de la littérature : + $\sim$ 16 pp en justesse. L'approche [Ryoo 2015] en comparaison, infère les classes d'actions seulement à partir de caractéristiques d'image 2D et du flot optique. Nous montrons que l'ajout d'information additionnelle à propos de la configuration spatiale des objets et leur nature enrichie les modèles d'actions, et donc les performances de reconnaissance d'actions.

L'approche KMHIS [Kataoka 2016] repose la reconnaissance de motifs de mouvement en



(a) CAD-120 (b) Watch-n-Patch

FIGURE 4.11 – Matrices de confusion, sans pré-segmentation des séquences d'action avec la vérité terrain, sur les deux jeux de données.

2D au travers d'image différentielles. Notre amélioration significative est aussi due à notre modélisation 3D de la scène, ce qui est plus robuste aux changements dans les points de vue des caméras présents dans ce jeu de données. En effet comme illustré sur la figure 4.9, les positions de l'acteur par rapport à la caméra sont variées.

Sur la figure 4.11b, nous observons de bonnes performances sur les actions les plus fréquentes dans le jeu de données. La distribution des classes est déséquilibrée : les actions lire et jouer à l'ordinateur représentent 50% des actions.

Alors que dans les deux cas la posture humaine est similaire : la personne es assise en face d'un bureau avec de légers mouvement au niveau des mains. Elles diffèrent grandement dans la nature des objets en interaction : un livre ou un clavier. Nous obtenons également de bonnes performances sur des actions moins fréquentes comme éteindre l'écran, qui ne représente que 4,27% des trames. Cependant, dans notre approche, les actions *put-back-book* et *put-down-item* sont sujettes à des erreurs dans leur reconnaissance à cause de la posture humaine qui est similaire et que le livre ou l'objet soit dans les deux cas sur la table. Nous devons modifier nos modèles pour que notre approche ait une meilleure spécificité dans la reconnaissance de l'action *put-down-item* par rapport aux autres actions et notamment *put-back-book* et *read*.

La figure 4.12 illustre le comportement de notre reconnaissance d'actions au long des trames d'une séquence de CAD-120 [Koppula 2013b]. Nous pouvons également noter que des actions diffèrent aussi selon leur durée d'exécution. Certaines sont typiquement plus longues comme déplacer et ouvrir en contraste avec les actions placer et atteindre. Nous avons des erreurs sur les transitions d'actions, sur cet exemple nous tendons à détecter l'action suivante comparée à la vérité terrain. À la fin de l'exécution de l'action, la probabilité du modèle décroît et la matrice

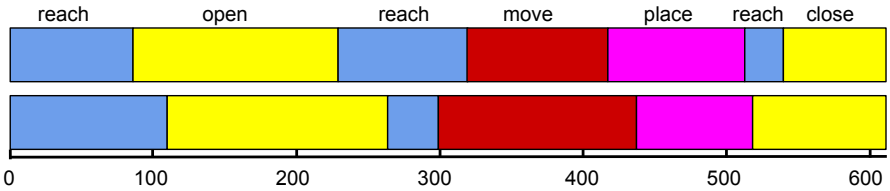


FIGURE 4.12 – Résultats quantitatifs sur la séquence réchauffer au micro-ondes (*microwaving food*) du jeu de données CAD-120. Les trames sont sur l'axe des abscisses. La première ligne est le résultat de notre approche et la deuxième correspond à la vérité terrain.

de transition favorise la probabilité de l'action suivante la plus probable. C'est particulièrement vrai pour CAD-120 où certaines actions sont déterminantes pour la suivante : verser et boire sont toujours suivies par déplacer.

4.5 Conclusion

Dans ce chapitre nous proposons une approche originale pour la reconnaissance d'actions en ligne à partir d'une modélisation conjointe de la posture et des objets observés dans le repère 3D. Les différents modèles sont décrits par des densités de probabilités gaussiennes dont les écarts-types sont appris via l'utilisation d'un outil d'optimisation des paramètres SMAC. Cela nous permet d'effectuer de la reconnaissance d'actions en ligne,

Notre mécanisme de modélisation a pour but de lever les ambiguïtés lors de la labelisation des actions, le déplacement d'un certain objet se retrouve dans différentes classes d'actions où la nature de l'objet ou son affordance renforce la prédiction de la bonne action. Nous n'avons pas besoin d'un grand nombre de vidéos annotées pour reconnaître une action similaire dans un nouveau jeu de données comme cela peut-être le cas, notamment pour les techniques d'apprentissage supervisées, surtout de types CNNs. Notre approche s'est montrée robuste aux variations de points de vue de la caméra grâce à la modélisation 3D de la scène.

Nos évaluations sur deux jeux de données de la littérature mettent en lumière une amélioration surtout dans le cas de la détection d'actions en ligne.

Ils ont permis la publication d'un article dans la conférence IEEE AVSS 2019. La référence complète se trouve en page vii.

CHAPITRE 5

Fusion de modèles bayésiens et de convolutions pour la reconnaissance d'actions

Sommaire

5.1 Introduction

Historiquement les approches probabilistes [Koppula 2013b] [Li 2010] proposent de caractériser les actions de manière explicite à travers une modélisation des observations des éléments inférées dans la scène. Si ces approches basées sur des modèles probabilistes offrent des performances généralement moindre que les réseaux de convolution, elles requièrent généralement une quantité de données d'apprentissage moins importante car elles ont aussi moins de paramètres sous-jacents à estimer ou régler. Leur interprétabilité est donc moins dépendante des données d'apprentissage (c-à-d moins sujet au sur-apprentissage). Ces approches sont donc pertinentes dans le cas d'un petit nombre d'échantillons disponibles pour un entraînement. Par exemple notre approche bayésienne, ANBM [Maurice 2019], décrite dans le chapitre précédent modélise à la fois les interactions entre objets, homme-objets à travers environ 50 paramètres. Notons que notre approche ANBM permet en plus de prendre en compte les transitions entre différentes actions afin d'assurer la cohérence temporelle durant l'exécution de séquences d'actions.

Des réseaux de neurones à convolution existent pour la reconnaissance d'actions et nous en citons quelques uns dans le chapitre 2. En raison du grand nombre de paramètres (13 millions) le réseau C3D requiert beaucoup données annotées pour être pertinent, l'apprentissage est notamment complexe sur les classes sous-représentées. Notre approche ANBM dépend des modèles qui ont été élaborés et même avec peu de données la prédiction des classes sous représentées est possible. Les deux approches, de part leur nature différente, sont à même d'offrir des performances complémentaires.

Forts de ces constats, nous proposons une approche qualifiée de IA hybride avec une fusion, au niveau décisionnel, d'un réseau de neurones à convolution de type C3D [Tran 2015] et de notre approche probabiliste ANBM [Maurice 2019] qui repose sur des percepts homme-objets explicites présentée dans le chapitre précédent. Ces deux approches doivent prendre en compte les caractéristiques spatio-temporelles des différentes classes d'actions au sein de l'exécution de l'action mais également dans leur séquencement logique.

Ainsi, nos contributions sont ici : (1) l'ajout d'une couche récurrente GRU (*Gated Recurrent Unit*) à C3D pour la reconnaissance d'actions afin de modéliser les corrélations temporelles entre actions successives (contribution mineure), (2) la comparaison de deux approches ANBM et C3D-GRU sur deux datasets publics CAD-120 et Watch-n-Patch, (3) la mise en œuvre et l'évaluation d'une fusion tardive de ces deux approches (donc fusionnant leurs prédictions) et comparaison avec la littérature sur ces deux datasets afin d'observer les plus-values de notre stratégie hybride avec fusion tardive.

Le chapitre s'organise comme suit. En section 5.2 nous présentons un réseau de convolutions 3D de la littérature ainsi que notre apport afin de prendre en compte la cohérence temporelle des actions. Puis dans la section 5.3 nous présentons notre approche hybride avec le mécanisme de fusion pour la détection d'actions. Une étude comparative de nos résultats est présentée dans la section 5.4. Enfin, la section 5.5 conclut sur ces investigations et énumère quelques perspectives.

5.2 Réseaux à convolutions en 3D

Nous décrivons ci-après en section 5.2.1 un réseau de neurones à convolution 3D de la littérature que nous privilégions : C3D [Tran 2015]. Puis dans la section 5.2.2 nous présentons l'ajout de la couche GRU afin de prendre en compte le contexte temporel dans lequel s'inscrit l'action avec les transitions entre actions : C3D-GRU.

5.2.1 Descriptif du réseau C3D

Le réseau C3D [Tran 2015] est un réseau d'apprentissage profond qui prend en compte une séquence d'images et donc la troisième dimension correspond ici au temps. L'introduction de filtres de convolution 3D permet donc d'apprendre des descripteurs spatio-temporaux à partir d'un flux vidéo. Ces réseaux apprennent de manière implicite des descripteurs liés au mouvement inter-images sur un horizon temporel de 16 trames classiquement. Les convolutions 3D prennent en compte le voisinage spatial d'un pixel donné mais également son voisinage tempo-

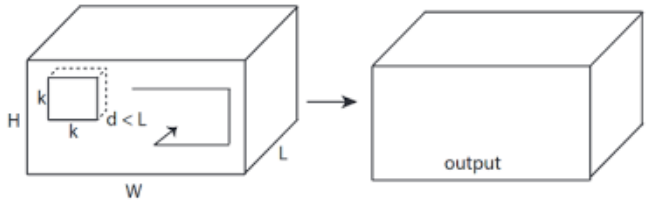


FIGURE 5.1 – Illustration de convolutions 3D sur une vidéo.

rel. Les convolutions 3D appliquées à une vidéo donnent en sortie un volume et non une image, préservant ainsi l'information temporelle contenue dans le signal en entrée comme illustré par la figure 5.1 issue de [Tran 2015]. Sur cette figure 5.1 H et W correspondent respectivement à la hauteur et la largeur d'une trame vidéo alors que L correspond aux nombre de trames. La taille du noyau de convolution y apparaît également : k $\times$ k $\times$ d. Les trames sont re-dimensionnées ou re-cadrées pour obtenir en entrée une trame de taille $112 \times 112 \times 3$, le nombre 3 correspondant au nombre de canaux couleurs.

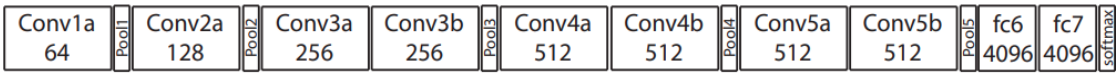


FIGURE 5.2 – Architecture du réseau C3D avec huit couches de convolutions, cinq couches de max pooling et deux couches denses.

L'architecture globale est organisée en couches et comprend des couches de convolutions (Conv) suivies par des couches de pooling (Pool) et pour finir deux couches densément connectées (Fc) et une opération de softmax pour prédire les actions comme nous pouvons le voir sur la figure 5.2 de et dans [Tran 2015]. Nous décrivons brièvement ces couches et la fonction softmax dans ce qui suit.

Couche de convolution - Ces couches appliquent une opération de convolution par un noyau de taille définie par la couche au signal d'entrée. Elles permettent au réseau d'apprendre des cartes de caractéristiques.

Couche de pooling - Le pooling est un processus de sous-échantillonnage. Son principal objectif est de réduire la dimensionnalité des caractéristiques d'entrée tout en conservant les informations les plus importantes. Le pooling est effectué en utilisant une fonction de pooling pour remplacer la sortie du réseau à un certain endroit par une statistique d'une zone correspondant à la taille du noyau de pooling. Par exemple la moyenne ou le maximum. Ici dans le réseau C3D toutes les couches de pooling utilisent la fonction maximum avec un noyau de taille $2 \times 2 \times 2$ (sauf pour la première couche) avec un pas de 1. Ainsi la taille de signal de sortie, par rapport au signal d'entrée, est réduite par un facteur 8.

Couche dense - Une couche entièrement connectée relient chaque neurone d'une couche à chaque neurone d'une autre couche. Tous les neurones sont connectés deux à deux.

Fonction softmax - Une fonction softmax est appliquée au vecteur obtenu à partir de la couche dense. En sortie nous obtenons un vecteur de valeurs positives dont la somme est égale à 1. Pour un vecteur $f = (f_1..., f_N)$ de N nombres réels, la fonction softmax est définie comme suit avec e la fonction exponentielle :

$$\text{softmax}(f)_i = \frac{\mathrm{e}^{f_i}}{\sum_{n=1}^{N} \mathrm{e}^{f_n}}. \tag{5.1}$$

Ainsi, D. Tran *et al.* proposent une description compacte, de taille 4096, d'un flux vidéo de taille $112 \times 112 \times 3 \times 16$. À l'instar des autres réseaux, C3D permet un apprentissage de bout en bout (*end-to-end*) sans requérir de lourds pré-traitements des vidéos par d'autres réseaux (OpenPose, SSD), ni d'informations d'experts, contrairement aux approches basées sur l'élaboration de modèles probabilistes telle que ANBM.

Par contre eu égard aux millions de paramètres à apprendre, les classes faiblement représentées sont plus difficiles à apprendre et à prédire correctement. L'action doit être échantillonnée sur une fenêtre fixe, 16 trames dans l'implémentation originale, bien sûr il est possible d'agrandir cette fenêtre temporelle mais cela requiert d'autant plus de mémoire et le choix de 16 trames apparaît comme un bon compromis dans la littérature. Tran *et al* dans [Tran 2015] proposent également un système de fenêtre glissante qui permet de moyenner les descripteurs au cours d'une action en considérant un horizon temporel plus long, par exemple 32 trames. Cependant, dans tous les cas, C3D présuppose un pré-découpage des actions dans les séquences, ce qui n'en fait pas une méthode adaptée pour de la reconnaissance d'action à la volée c-à-d sans segmentation préalable des actions. De fait de nombreux challenges et jeux de données associés pour la reconnaissance d'actions sont construits dans cette perspective : Sports-1M [Karpathy 2014], YouTube-8M [Abu-El-Haija 2016]. Les actions n'y sont pas localisées dans le temps comme souligné dans le chapitre 2.

De plus il n'y a pas de mécanisme pour prendre en compte lors de l'apprentissage le contexte temporel dans lequel le clip vidéo s'insère. Il n'y a donc pas de prise en compte de la nature de l'action précédente. En effet lors de l'entraînement les clips traités de façon indépendantes alors que dans une activité il y a un séquencement logique des actions et il est donc intéressant d'ajouter un mécanisme pour prendre en compte cette information liée de temporalité.

5.2.2 Ajout d'une couche récurrente : C3D-GRU

Afin de pallier l'absence de mécanisme de prise en compte de corrélation avec l'action précédente, nous proposons après entraînement de C3D, de récupérer les poids du modèle entraîné, de les geler et d'y ajouter une couche récurrente de type réseau récurrent à portes GRU(*Gated Reccurent Unit*) [Cho 2014]. Les couches récurrentes sont faites pour obtenir des prédictions à partir de séquences de données. Il existe d'autres couches récurrentes telles que les couches LSTM (*Long Shorm Term Memory*), mais les LSTM ont plus de paramètres. En général les

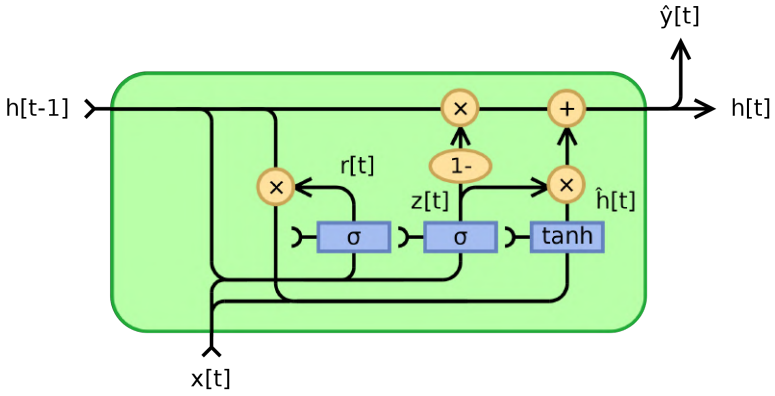


FIGURE 5.3 – Architecture d'un réseau récurrent à portes.

LSTM sont plutôt indiqués pour de longues séquences. Or ici nous sommes en présence d'une séquence courte c-à-d deux actions comme justifié ci-après. La couche GRU est plus récente, elle requiert moins de paramètres et est adaptée à notre séquence. Une unité GRU contient un état caché qui agit comme la mémoire du réseau.

Cette couche GRU est illustrée sur la figure 5.3, où h représente l'état caché, σ représente la fonction sigmoïd. Nous définissions les variables suivantes : le vecteur d'entrée x_t, la prédiction $\hat{y}$ à partir de l'historique h, W, U, b des matrices et vecteur de paramètres. La porte de mise à jour z est définie comme suit à l'instant t :

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z). \tag{5.2}$$

La porte de reset r à l'instant t est définie comme suit :

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r). \tag{5.3}$$

Et la sortie h, avec $\odot$ est le produit par élément, par :

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h). \tag{5.4}$$

Ensuite l'entraînement de ce réseau modifié diffère de celui qui est fait pour entraîné C3D. En effet, nous l'avons adapté afin de prendre en compte deux clips successifs en entrée qui correspondent à deux actions successives. Le choix d'utiliser deux clips successifs et non davantage nous permet de rester homogène vis-à-vis du modèle ANBM qui ne considère lui aussi que les transitions entre deux actions. Ce n'est possible que lorsque l'on dispose d'actions localisées temporellement et agencées en séquences ce qui est généralement le cas pour le suivi d'activités domestiques. Nous ne ré-entraînons pas l'ensemble du réseau C3D, nous faisons seulement un paramétrage fin (*fine-tuning*) au niveau de la couche GRU et de la dernière couche dense avec soft-max.

Pour illustrer l'importance du séquencement des actions, parmi l'ensemble des transitions

TABLE 5.1 – Statistiques sur le séquencement entre deux actions successives dans le jeu de données Watch-and-Patch environnement *office* .

	lire	marcher	sortir du bureau	attraper un livre	reposer le livre	poser un objet	prendre un objet	jouer à l'ordinateur	allumer l'écran	éteindre l'écran
lire	0	1	53	6	13	15	12	0	0	0
marcher	20	0	0	11	3	53	2	5	5	0
sortir du bureau	0	9	2	0	41	4	39	0	0	4
attraper un livre	76	21	0	0	0	3	0	0	0	0
reposer le livre	0	10	60	0	0	0	31	0	0	0
poser un objet	30	6	9	17	0	0	7	24	7	0
prendre un objet	0	1	94	0	0	4	0	0	0	1
jouer à l'ordinateur	0	0	13	0	0	1	15	0	38	33
allumer l'écran	0	0	0	0	0	7	0	93	0	0
éteindre l'écran	0	0	51	0	0	0	49	0	0	0

envisageables entre deux paires d'actions dans Watch-n-Patch [Wu 2015b], seules environ 45 % sont possibles comme le montre le tableau 5.1. Avec seulement 32 % des transitions sont fréquentes à plus de 5% dans le jeu de données. L'importance du séquencement des actions se vérifie également sur le jeu de données CAD-120 avec le tableau 4.2 du chapitre 4.

Cette stratégie est illustrée par l'architecture numéro 3 de la figure 5.4. Cette approche est notée C3D-GRU par la suite.

5.3 Fusion tardive avec couche dense

Nous proposons donc deux stratégies pour prédire les actions sur des clips vidéos à partir des données spatio-temporelles, de manière explicite avec ANBM et de manière implicite avec C3D. De plus ANBM modélise également les transitions qui existent entre deux actions successives. D'un coté dans ANBM nous avons modélisé chacune des actions et leurs transitions, de l'autre C3D-GRU va les apprendre à partir des données brutes, dont les classes ne sont pas réparties équitablement. En effet dans la détection d'actions atomiques, certaines actions se retrouvent plus fréquemment par exemple le déplacement d'un objet représente 34% des actions de CAD-120. Nous proposons ici la fusion de leurs prédictions respectives, donc une fusion dite tardive. En effet les deux approches prédisent un vecteur de probabilités de chacune des classes pour ANBM, et pour C3D-GRU nous avons un vecteur dont les valeurs sont comprises entre 0 et 1 et qui correspond à la sortie de la couche soft-max.

Nous proposons une stratégie qui prend en entrée les clips vidéos qui passent à travers l'approche ANBM et également à travers le réseau C3D-GRU décrit précédemment. L'ensemble des poids du réseau C3D-GRU sont maintenant gelés. Nous obtenons donc deux vecteurs de prédictions pour chacune des approches que nous concaténons. Cette couche de concaténation est connectée à une couche dense de même taille que le nombre de classes : N. Il n'y a donc

que $N^2 + N$ paramètres à apprendre : N^2 poids liés à la couche dense et N biais liés à l'activation. La fonction d'activation retenue pour ce problème de classification multi-classes est la fonction soft-max. L'approche numéro 4 de la figure 5.4 en est une illustration, la couche dense est illustrée avec $N = 4$. Les prédictions de ANBM et de C3D-GRU sont inter-connectés, ce qui nous permet de tirer parti des deux approches dans la prédiction finale. Cette approche sera notée ANBM-C3D-GRU ci-après.

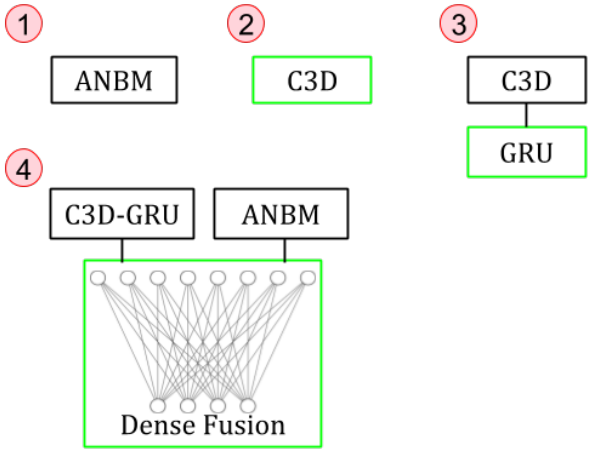


FIGURE 5.4 – Récapitulatif des différentes architectures individuelles (1) (2) (3) et leur fusion tardive au moyen d'une couche dense (4). L'ensemble des couches entraînées est entouré en vert.

5.4 Évaluations

5.4.1 Protocole expérimental

Jeux de données Les différentes approches sont évaluées sur deux jeux de données publics présentés dans le chapitre 1 : CAD-120 et Watch-n-Patch. Nous rappelons que ces deux jeux offrent des séquences d'actions et donc la possibilité d'évaluer l'intérêt de la cohérence temporelle entre actions successives dans la performance globale de reconnaissance d'action. De plus ils présentent des caractéristiques intéressantes comme le fait que les actions ne soient pas réparties de manière équilibrée comme le montre le tableau 5.2, ce qui constitue généralement un challenge pour les approches basées CNN comme en atteste le nombre d'investigations afin d'y remédier : augmentation de données, ajout de poids différents pour le calcul de la fonction de perte (*loss*), etc.

(a) Image originale

(b) Image découpée autour de la boîte englobante

FIGURE 5.5 – Pré-traitement des images pour l'entraînement de C3D. Images extraites du jeu de données CAD-120.

TABLE 5.2 – Détail de la répartition des classes et nombres de clips (Nb clips) dans les jeux de données.

Jeux de données	Nb clips	Répartition des classes (%)
CAD-120	1149	[23,30,3,3,3,15,4,3,1,14]
Watch-n-Patch (office)	1148	[12,16,21,6,4,14,9,9,5,3]

Gestion des prédictions de ANBM Pour rappel, nous avons enregistré les probabilités en sortie de ANBM, puis à travers un mécanisme de vote sur la durée de chacune des actions nous obtenons un vecteur de probabilité de prédiction par clip vidéo représentant une action.

Pré-traitements pour C3D Nous avons conservé les paramètres originaux de [Tran 2015] pour la taille des images en entrée en la fixant à 112×112. Les clips vidéos sont recadrés autour de la boîte englobante élargie contenant l'acteur et les objets du contexte de l'action comme illustré figure 5.5. Cette boîte englobante est définie au moyen de OpenPose [Cao 2017]. Ce qui permet de concentrer l'attention sur la zone où à lieu l'activité. Pour rappel, le réseau C3D prend une séquence d'action de taille fixe : 16 trames. En pratique, puisque l'on considère des actions atomiques, relativement courtes, nous n'utilisons pas une fenêtre glissante sur les séquences, mais plutôt simplement un ré-échantillonage des séquences.

Apprentissage Les poids du réseau sont entraînés en utilisant une descente de gradient stochastique sur des mini-lots de taille 16 avec un momentum de 0.9. Nous initialisons le taux d'apprentissage à 0.01 et il diminue par palier, il est divisé par 10 après 20 époques. Nous utilisons la fonction de coût *categorical cross-entropy* car nous avons une seule classe d'action par segment vidéo, les actions ne sont pas concomitantes. L'entraînement est fait sur une carte graphique GeForce GTX 1080 Ti.

Test Les performances de notre approche hybride sont évaluées suivant le principe de la validation croisée de type *k-fold* avec $k = 4$. Chacun des échantillons forment une partition de l'ensemble des données du dataset. Chaque sous-échantillon est utilisé exactement une fois comme ensemble de validation lors de l'entraînement. Dans le jeu de données CAD-120 il y a quatre acteurs et chaque échantillon est associé à un acteur. Dans Watch-n-Patch la publication originale [Wu 2015b] fournit un seul ensemble de test et d'entraînement, nous en avons généré 3 autres en prenant soin de garder les actions d'une même séquence au sein du même pli (*fold*). Nous obtenons la prédiction finale au niveau de la dernière couche d'activation (soft-max) présente dans les approches 2,3,4 décrites dans le chapitre 3 et les sections 5.2 5.3 et illustrées figure 5.4. Cette couche d'activation soft-max de la même taille que le nombre de classes permet d'obtenir une normalisation des valeurs brutes des couches de neurones entre 0 et 1. La somme des valeurs obtenues pour chacune des classes vaut 1. La fonction soft-max prend en entrée un vecteur $z = (z_1, ..., z_k)$ et est définie comme suit :

$$\sigma(z)_j = \frac{e^{z_j}}{\sum e^{z_k}}, j \in \{1, ..., K\} \tag{5.5}$$

Métriques pour évaluation Nous évaluons les différentes variantes énumérées dans la section 5.3 via deux métriques abondamment référencées dans la littérature. La première la justesse, dénommée micro-justesse et notée μ par la suite, est définie comme suit :

$$\text{Justesse} = \frac{\text{Nombre de prédictions correctes}}{\text{Nombre total de prédictions}} \tag{5.6}$$

qui est le ratio des actions correctement reconnues par nombre total d'actions à reconnaître. Nous l'appelons ainsi par opposition avec la macro-justesse. La macro-justesse, notée M, est la moyenne des justesses pour chacune des classes.

La macro-justesse donne le même poids à chacune des classes, indépendemment du nombre d'échantillons de cette classe dans le jeu de données. Cela permet de voir si ce sont seulement les classes les plus représentées qui sont correctement reconnues où si globalement l'ensemble des classes, y compris celles sous-représentées sont reconnues correctement. Ces deux métriques sont complémentaires dans l'évaluation des performances pour des jeux de données dont la répartition des classes n'est pas équilibrée.

5.4.2 Résultats et discussion

Nous allons d'abord comparer individuellement les résultats des approches C3D, C3D-GRU et ANBM décrites en sections 5.2 et chapitre 3 avant de quantifier les gains lors leur fusion.

Tout d'abord nous évaluons l'apport de la couche récurrente GRU à la sortie de C3D pour prendre en compte la logique entre les actions. D'après le tableau 5.3 sur Watch-n-Patch nous observons en moyenne un gain en micro-justesse de +13 points de pourcentage (pp) et d'après le tableau 5.4 un gain +4 pp sur CAD-120 grâce à l'ajout d'une couche GRU au réseau C3D par rapport à C3D seul. En regardant en détail les matrices de confusion des figures 5.6a et 5.6b

TABLE 5.3 – Résultats de nos différentes approches sur Watch-n-Patch. Les performances considérées sont les macro-justesse (M) et la micro-justesse (μ).

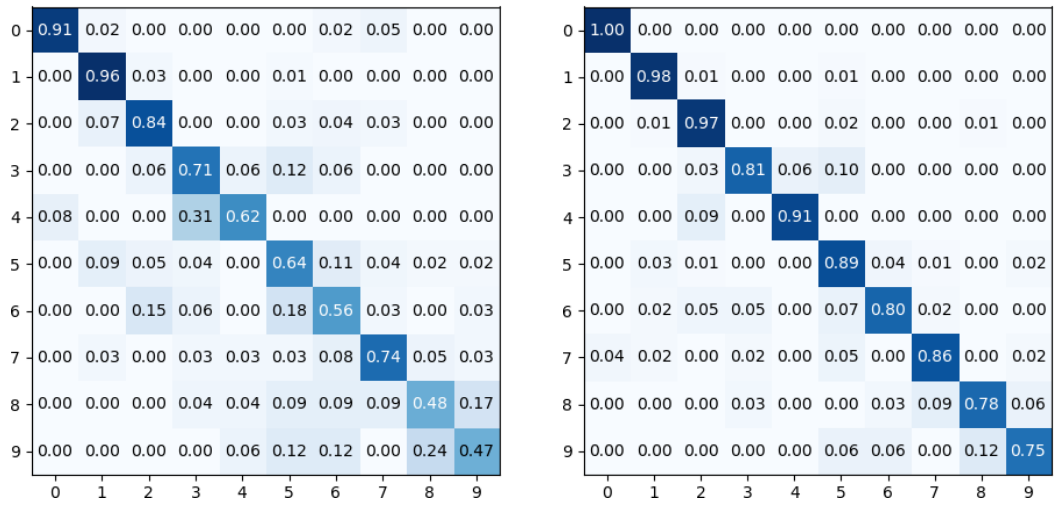
Approche	Pli 0		Pli 1		Pli 2		Pli 3		Moyenne		Écart-type	
	μ	M	μ	M	μ	M	μ	M	μ	M	μ	M
ANBM	78	79	73	74	76	77	75	75	76	76	2	2
C3D	72	65	73	64	75	69	74	64	74	66	1	2
C3D-GRU	89	87	86	77	85	77	89	84	87	81	2	5
ANBM-C3D-GRU	**94**	**91**	**93**	**90**	**93**	**91**	**93**	**89**	**93**	**90**	**0.1**	**1**

TABLE 5.4 – Résultats quantitatifs des différentes approches sur CAD-120. Les mesures considérées ici sont la macro-justesse (M) et la micro-justesse (μ).

Approche	A1		A2		A3		A4		Moyenne		Écart-type	
	μ	M	μ	M	μ	M	μ	M	μ	M	μ	M
ANBM	84	77	78	81	82	76	82	77	82	78	3	**2**
C3D	58	45	70	61	64	57	56	35	62	50	6	12
C3D-GRU	61	49	76	73	66	60	60	45	66	57	7	13
ANBM-C3D-GRU	**86**	**80**	**89**	**91**	**84**	**82**	**83**	**79**	**86**	**83**	**3**	5

nous observons que, sur le jeu de données Watch-n-Patch, les classes qui en ont le plus bénéficié sont les classes suivantes : *remettre un livre*, *poser un objet* et *prendre un objet*. En effet l'action *remettre un livre* est souvent précédée de l'action *lire*. Lorsque l'action précédente détectée est *lire* cela réduit et conditionne le choix des possibilités suivantes. L'action de *poser un objet* est elle fréquemment précédée par l'action *marcher*. En effet dans Watch-n-Patch il est fréquent qu'une personne entre dans le bureau en marchant et dépose son téléphone sur la table. Les gains sur CAD-120 sont plus modestes car pour que la couche récurrente apporte de l'information il faut que le réseau C3D gelé ait suffisamment bien appris au préalable à reconnaître les classes.

FIGURE 5.6 – Matrices de confusion de C3D et C3D-GRU sur le jeu de données Watch-n-Patch. Les prédictions apparaissent sur les colonnes et la vérité terrain sur les lignes.



(a) C3D (b) C3D-GRU

Le réseau C3D-GRU surpasse donc C3D ; comparons alors avec notre approche ANBM avant d'évaluer leur fusion. Sur le jeu de données Watch-n-Patch, C3D-GRU a une meilleure micro-justesse que ANBM avec un écart de +11 pp mais l'amélioration de la macro-justesse est moins importante avec +5 pp, cf. lignes 1 et 3 du tableau 5.3. Comme observé sur la matrice de confusion de la figure 5.6b, les actions les mieux détectées sont lire, marcher et sortir avec des scores respectivement de 1, 0,98, et 0,97. Ces actions représentent 49% des données ($12 + 16 + 21 = 49$ cf. tableau 5.2) et contribuent plus à la micro-justesse que éteindre l'écran qui ne représente que 3% de l'ensemble des séquences. La matrice de confusion de la figure 5.8a nous montre que l'approche ANBM surpasse C3D-GRU sur 3 classes : jouer à l'ordinateur, allumer l'écran, allumer l'écran et éteindre l'écran. Les deux approches offrent des approches complémentaires sur des classes différentes et leurs sources de confusions varient également. Comme le montre les matrices de confusion des figures 5.8a et 5.8b, les deux approches ont des performances similaires pour l'action attraper un livre (0,71 pour ANBM et 0.81 pour C3D-GRU) mais les erreurs diffèrent. En effet, ANBM détecte parfois prendre un objet alors que C3D-GRU détecte à la place l'action poser un objet. Sur CAD-120 le réseau C3D-GRU distingue mieux atteindre et placer que ANBM, ces deux actions représentent 38% du jeu de données ($23 + 15 = 38$ cf. tableau 5.2).

Sur les deux jeux de données, les deux approches qui prennent en compte la cohérence temporelle entre les actions : C3D-GRU et ANBM apportent donc des performances complémentaires. Nous évaluons ici les bénéfices que l'on peut tirer de leur fusion. Sur CAD-120, la capacité pour le réseau C3D-GRU à distinguer *atteindre* et *placer* des autres classes permet lors de la fusion des deux approches d'avoir des gains de +4 pp en micro-justesse, voir tableau 5.4. La fusion de C3D-GRU et ANBM a permis d'améliorer la reconnaissance de toutes les actions

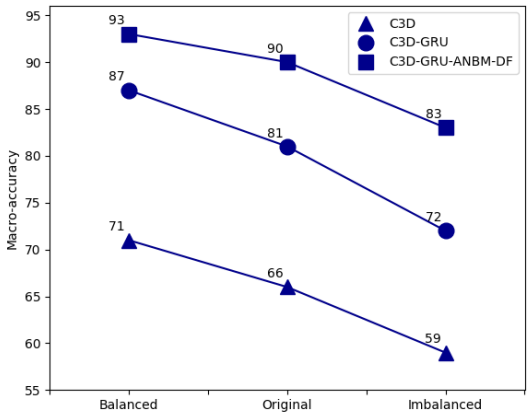


FIGURE 5.7 – Macro-justesse en fonction du déséquilibre des classes sur Watch-n-Patch. Les classes sont synthétiquement augmentées ou dégradées.

de Watch-n-Patch sauf de l'action *marcher* qui passe de 0.98 avec C3D-GRU à 0.97 comme le montre les matrices de confusion sur les figures 5.8a 5.6b et 5.8c. Dans l'ensemble, la fusion permet d'augmenter la micro-justesse de +6 pp par rapport à C3D-GRU et de +17 pp par rapport à ANBM. Pour les actions mettant en jeu un ordinateur, dans la fusion ce sont les performances de ANBM qui sont privilégiées par rapport à celles de C3D-GRU. La complémentarité des deux approches est bien exploitée également lorsqu'ils présentent individuellement performances similaires pour une classe d'action. Par exemple dans la fusion l'action *attraper un livre* atteint 0,94.

Nous proposons alors d'évaluer la robustesse de notre approche de fusion sur le jeu de données Watch-n-Patch en égalisant ou en dégradant le déséquilibre des classes. De manière artificielle nous augmentons ou dégradons le nombre d'échantillons des classes, puis nous re-entraînons les réseaux C3D, C3D-GRU et C3D-GRU-ANBM-FD pour obtenir les résultats présentés sur la figure 5.7. Nous notons que l'entraînement de C3D est sensible au nombre d'échantillons. Observons également la dépendance de C3D-GRU aux résultats obtenus par C3D seul. En effet, la performance de C3D-GRU décroît plus rapidement que celle de C3D, car pour capturer la cohérence temporelle, l'action précédente doit être bien détectée. Quand les classes sont fortement déséquilibrées, C3D détecte mal certaines actions et certaines transitions entre actions ne sont pas modélisées. Dans l'ensemble notons que la fusion C3D-GRU-ANBM-FD est plus robuste à la dégradation du nombre d'échantillons.

Comme le montre le tableau 5.5, notre stratégie de fusion permet d'améliorer nos performances précédentes, tout en ayant des performances proches ou supérieures à celles de l'état de l'art. Nous avons suivi le protocole d'évaluation suggéré par les auteurs de CAD-120, il est à noté que GPNN [Qi 2018a] s'est évalué dans un *setup* différent. Nous avons choisi des approches récentes de référence de l'état de l'art, si possible s'évaluant sur les deux mêmes jeux

TABLE 5.5 – Comparaison à la littérature. Justesse de la détection des actions sur CAD-120 et Watch-n-Patch.

Jeu de donnée	Approche	Justesse
CAD-120	GEPHAPP [qi2]	79.4
	ANBM [Maurice 2019]	82.2
	GPNN [Qi 2018a]	87.3
	Notre approche	**86.1**
Watch-n-Patch	WBTM [Wu 2016]	35.2
	PoT [Ryoo 2015]	49.93
	ANBM [Maurice 2019]	76.4
	GEPHAPP [qi2]	84.8
	Notre approche	**93.0**

de données telle que GEPHAPP [qi2]. Les deux approches considérées dans notre stratégie de fusion prennent en compte les transitions d'une action précédente vers une suivante. Or dans CAD-120, l'action *déplacer* précède presque toutes les autres, ce qui est peu déterminant. On y voit également que notre stratégie de fusion permet de surpasser l'état de l'art en reconnaissance d'actions sur le jeu de données Watch-n-Patch en améliorant la reconnaissance d'actions de +8,4 pp par rapport à l'approche proposée par Qi *et al* [qi2]. Une vidéo qui illustre nos résultats sur les séquences vidéos est disponible à l'adresse indiquée en pied de page [1].

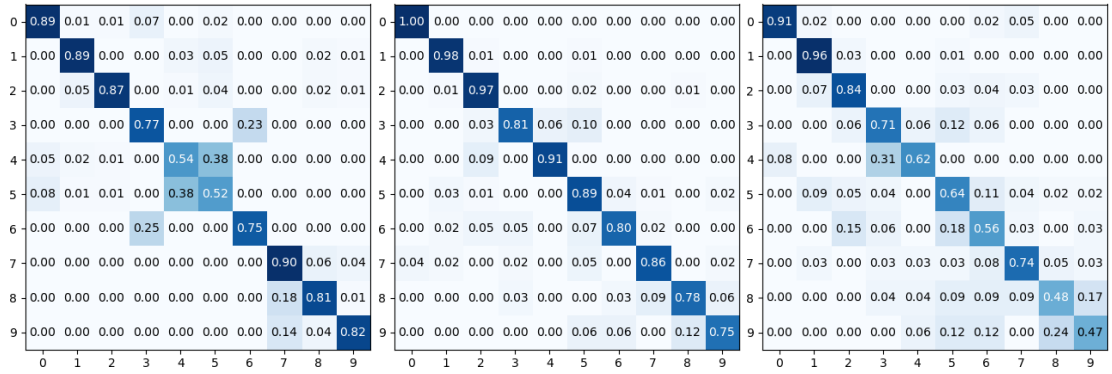
5.5 Conclusion

Dans cet article nous avons comparé différentes approches de détection d'actions et proposé l'ajout de couche récurrente à C3D pour tirer parti des relations temporelles qui existent entre actions successives. Nous avons exploré une manière de fusionner au niveau décisionnel deux approches de reconnaissance d'actions au moyen d'une couche dense. Nous les avons expérimentées sur deux jeux de données de la littérature présentant un déséquilibre entre les classes et nous avons montré des gains d'autant plus significatifs que les approches sont complémentaires. Les réseaux de neurones à convolution n'ont pas systématiquement de meilleures performances, une approche paramétrique avec un bon réglage des paramètres avec un outil d'optimisation peuvent être efficaces.

Ce mécanisme de fusion permet de tirer parti d'approches complémentaires et d'obtenir des gains en performances, en revanche la mise en œuvre de plusieurs approches dont un réseau deep-learning demande beaucoup de temps et de ressources. C'est pourquoi dans dans le chapitre suivant nous allons nous intéresser à des réseaux plus compacts : les réseaux à convolution sur des graphes.

1. https ://youtu.be/Z-cgTcZvRiY

FIGURE 5.8 – Matrices de confusion de la reconnaissance d'actions sur le jeu de données Watch-n-Patch par les différentes approches. Les prédictions apparaissent sur les colonnes et la vérité terrain sur les lignes. [0 - lire ; 1 - marcher ; 2 - sortir du bureau ; 3 - attraper un livre ; 4 - reposer un livre ; 5 - poser un objet ; 6 - prendre un objet ; 7 - jouer à l'ordinateur ; 8 - allumer l'ordinateur ; 9 - éteindre l'ordinateur].



(a) ANBM (b) C3D GRU (c) Fusion

Les travaux présentés dans ce chapitre ont permis la publication de deux articles de conférences : RFIAP 2020 et IEEE ICPR 2020. Les références complètes se trouvent en page vii.

CHAPITRE 6

Reconnaissance d'actions par réseaux à convolutions pour les graphes

Sommaire

6.1 Introduction

Comme indiqué au chapitre 5, les approches reposant sur des architectures de type réseau de neurones à convolution (CNN) adaptées au domaine vidéo avec des convolutions 3D permettent la reconnaissance des actions dans les flux vidéo. Les réseaux de neurones à convolution 3D apprennent simultanément des caractéristiques spatio-temporelles. Des approches comme C3D [Tran 2015] obtiennent une précision de 90,4% pour le jeu de données de reconnaissance d'actions UCF101 [Soomro 2012b]. Cependant, l'emploi de convolutions 3D augmente la taille du réseau et donc le nombre de paramètres à apprendre (c'est-à-dire 15 millions avec C3D). Elles reposent souvent sur de grands jeux de données pour l'apprentissage, tels que Kinetics [Kay 2017] et UCF101 qui sont créés à partir de vidéos collectées sur YouTube. Les différentes actions sont exécutées dans des environnements radicalement différents, avec peu ou pas d'objets présents dans la scène. Par exemple, pour l'action *nager* aucun objet n'est présent et dans *jouer de la guitare* seule la guitare est présente dans cette séquence et la guitare n'apparaît que dans les séquences de cette action. Ces méthodes ne conviennent pas à la reconnaissance d'actions de la vie quotidienne où les mêmes objets dans le même environnement sont utilisés pour effectuer des actions différentes.

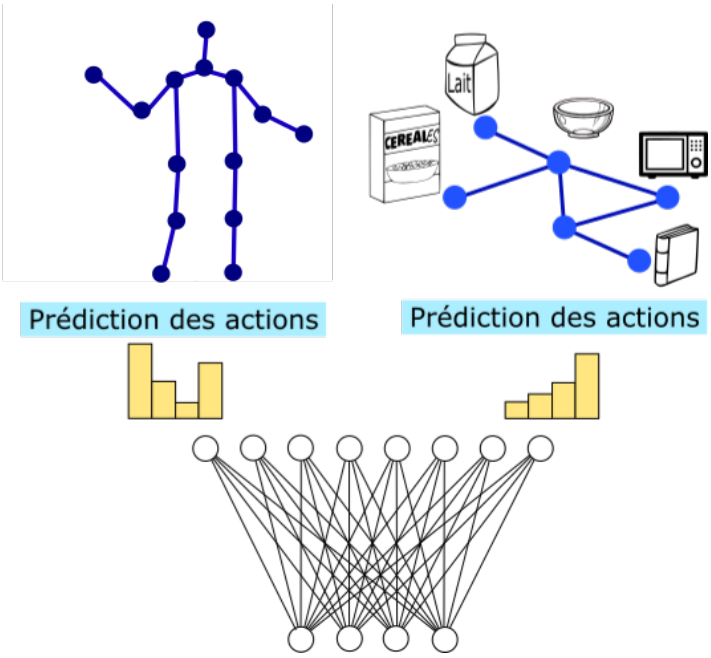


FIGURE 6.1 – De gauche à droite : le graphe-squelette et le graphe-objet, dans un soucis de clarté seules les connexions spatiales sont représentées ici. Leurs prédictions sont concaténées et connectées via une couche dense pour fusionner tardive.

Les progrès réalisés dans l'estimation de la posture humaine avec le célèbre OpenPose et dans la reconnaissance des objets nous permettent d'extraire à partir de flux vidéo les trajectoires au cours du temps des articulations du squelette ainsi que la nature et les positions des objets. Nous estimons que ce sont des informations pertinentes qui permettent de discriminer des actions. Ainsi, dans ce chapitre nous proposons de considérer à nouveau ces percepts et de les modéliser au travers de deux graphes et d'en évaluer la plus-value éventuelle. Depuis l'introduction des réseaux à convolution pour graphes (GCN) par Kipf et Welling [Kipf 2017] en 2017, où les auteurs proposent une généralisation des réseaux neuronaux à convolution à toute structure de graphe arbitrairement fixée. Les GCN ont gagné en popularité au sein de la communauté Vision par Ordinateur. En particulier dans le domaine de la reconnaissance d'actions où les articulations et les membres du squelette humain définissent naturellement un graphe comme illustré en haut-gauche de la figure 6.1. En sus du graphe sur le squelette, l'originalité est de modéliser les relations entre divers objets par un graphe, dont les nœuds portent des caractérisques liées à la nature et la position des objets et où un arc existe entre deux nœuds s'ils peuvent interagir. Nous sommes capables de modéliser ces séquences temporelles de positions 2D dans l'image sous forme de graphes spatio-temporels et d'obtenir une reconnaissance d'actions via GCN.

Fort de ces constats, nous proposons une architecture modulaire de reconnaissance des actions basé sur les GCN. Nous proposons l'utilisation d'un GCN basée sur un graphe squelette

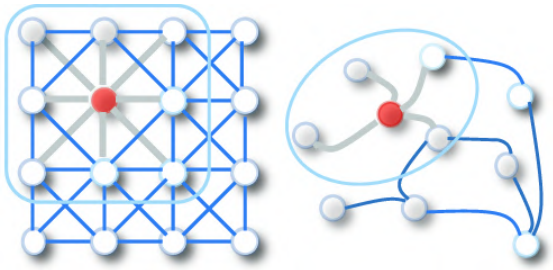


FIGURE 6.2 – À gauche illustration de l'opération de convolution sur une image, à droite illustration sur un graphe.

temporel similaire à [Yan 2018], en outre nous proposons d'ajouter les interactions des objets dans un GCN séparé. Ce cadre modulaire met en lumière l'importance des interactions entre objets pour lever les ambiguïtés dans la prédiction de l'action humaine à partir du GCN squelette. Nos contributions sont donc les suivantes : (1) La conception d'un graphe d'interactions entre objets pour modéliser les interactions en plus d'un graphe basé sur le squelette (2) la comparaison des performances des modèles de graphes-squelettes et de graphes-objets sur trois jeux de données publics de la littérature avec de riches interactions entre objets : CAD-120, Watch-n-Patch et Charades (3) le mécanisme de fusion tardive présenté au chapitre 5 cette fois mis en œuvre afin d'obtenir une prédiction finale en combinant les prédictions des graphes squelette et objet. En outre, par rapport à d'autres approches de la littérature par exemple [Qi 2018b], notre modèle est 15 fois plus compact vis-à-vis du nombre de paramètres.

Le chapitre s'organise comme suit : dans la section 6.2 nous présentons notre approche modulaire pour la reconnaissance d'actions. Ensuite nous détaillons le protocole d'évaluation, nos résultats et une étude comparative dans la section 6.3. Enfin, la section 6.4 présente notre conclusion et énonce quelques perspectives.

6.2 Réseaux à convolution pour les graphes

Dans cette section, nous commençons par rappeler le concept de convolution sur les graphes. Puis nous évoquons les réseaux à convolution spatio-temporels appliqué aux trajectoires prises par la posture humaine lors de l'exécution d'une action comme dans [Yan 2018]. Ensuite, nous décrivons notre extension pour prendre en compte la configuration des objets au travers de leurs trajectoires et interactions. Afin de prendre en compte les prédictions de ces deux modèles, nous proposons alors une stratégie de fusion tardive inspirée du chapitre 5.

6.2.1 Convolutions et graphes

Les réseaux à convolution pour les graphes introduits par Kipf et Welling [Kipf 2017] appliquent des convolutions aux graphes au lieu d'une image. L'image est composée de pixels disposés dans une grille, et la convolution opère sur les pixels voisins de manière ordonnée.

Contrairement aux convolutions sur les images, pour un nœud du graphe, le nombre de voisins varie et n'est pas ordonné. Comme nous pouvons le voir sur la figure 6.2 ou sur la figure 6.3 où le nœud correspondant au coude a deux voisins spatiaux et un temporel, alors que le nœud correspondant à la cheville a un voisin temporel mais qu'un de nature spatiale qui correspond au genou.

Les convolutions appliquées aux graphes nous permettent de calculer la réponse d'un nœud en fonction de ses voisins définis par les relations d'adjacences du graphe. Les convolutions sur les images considèrent celle-ci comme une grille 2D et permettent d'obtenir une carte des caractéristiques également sous forme de grille 2D. Avec un pas de 1 et un *padding* approprié nous supposons à l'instar de [Yan 2018] que la carte de caractéristiques obtenue fait la même taille que l'entrée.

Étant donné un opérateur de convolution avec un noyau de taille $k \times k$ et une carte des caractéristiques f_{in} avec c canaux. La valeur de sortie pour un seul canal à l'emplacement x peut être définie par :

$$f_{out}(x) = \sum_{h=1}^{K} \sum_{w=1}^{K} f_{in}(\mathbf{p}(x,h,w)) \cdot \mathbf{w}(h,w), \tag{6.1}$$

où p est une fonction d'échantillonage qui énumère les localisations des voisins de x dont l'ensemble de départ est $\mathbb{Z}^2 \times \mathbb{Z}^2$ et l'ensemble d'arrivée $\mathbb{Z}^2$. Et w la fonction de poids. Dans les convolutions standards sur image les positions de $p(x)$ forment une grille.

L'opération de convolution sur les graphes est alors définie en re-définissant les fonctions d'échantillonage et de poids qui apparaissent dans l'équation 6.1. Sur les graphes la fonction p considère un nombre variables de voisins qui sont les nœuds adjacents du nœud considéré. Un des problèmes dans la redéfinition de la fonction de poids c'est que dans les graphes il n'y a pas une grille définie et un ordre spatial naturel autour du nœud considéré. Une solution est proposée par Niepert *et al.* [Niepert 2016], ils proposent de définir un ordre via une fonction d'étiquetage des noeuds voisins l. Cette fonction d'étiquetage l a pour ensemble de départ l'ensemble des noeuds d'un graphe et comme ensemble d'arrivée un ensemble ordonné comme des nombres entiers par exemple. Une illustration de la différence entre ces deux types de convolutions est sur la figure 6.2 et dans [Wu 2020] où le pixel ou le nœud considéré apparaît en rouge et le voisinage considéré et entouré en bleu ciel.

Grâce à ces adaptations, soit un graphe $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, avec $\mathcal{V}$ l'ensemble des nœuds et $\mathcal{E}$ l'ensemble des arêtes, $v_i \in \mathcal{V}$ et $\mathcal{A}(v_i)$ l'ensemble des voisins de v_i, l'opération de convolution peut-s'écrire :

$$f_{out}(v_i) = \sum_{v_j \in \mathcal{A}(v_i)} = \frac{1}{Z_i(v_j)} f_i n(v_j) \cdot w(l_i(v_j)). \tag{6.2}$$

Plus de détails sur cette formalisation se trouvent dans [Dai 2017, Yan 2018].

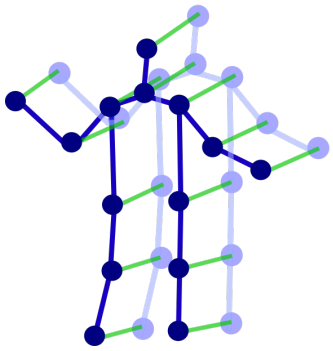


FIGURE 6.3 – Graphe spatio-temporel du squelette pour une séquence de deux trames à $t-1$ et t. Les arcs bleus (resp. verts) sont les arcs spatiaux (resp. temporels).

6.2.2 Réseau à convolution graphe-squelette

Suite à cette adaptation de CNN à GCN, un réseau de convolution de graphe spatio-temporel pour la reconnaissance d'actions a été introduit par S. Yan *et al.* [Yan 2018]. Ils modélisent uniquement le squelette humain comme un graphe spatio-temporel. Cette modélisation est motivée par le fait que les mouvements corporels sont très caractéristiques pour certaines actions notamment celles des jeux de données Kinetics [Carreira 2019] et NTU-RGBD [Shahroudy 2016] sur lesquels ils s'évaluent. Ce modèle est illustré par la figure 6.3. Le graphe du squelette est défini par un ensemble de nœuds et d'arêtes $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$. L'ensemble $\mathcal{V}_s$ comprend toutes les articulations d'une séquence du squelette. Chaque nœud est associé à un vecteur de caractéristiques (*features*) qui contient les positions 2D de l'articulation et un indice de confiance lié au détecteur. Le nombre de nœuds à un temps t dépend de la représentation du squelette utilisé, si nous utilisons celui fourni par OpenPose nous avons 18 articulations donc 18 nœuds. Deux nœuds sont connectés suivant les connexions naturelles du squelette humain.

6.2.3 Réseau à convolution graphe-objet

Nous nous intéressons à la reconnaissance d'actions lorsqu'une personne interagit avec des objets à proximité. Notre travail est basé sur leur publication avec leur graphe original du squelette spatio-temporel [Yan 2018]. Ce graphe est capable de modéliser des actions où la posture humaine est discriminante. À l'instar de nos travaux précédents, nous considérons que les objets sont des données contextuelles discriminantes.

Nous proposons donc de construire un second réseau à convolution pour graphes qui apprend à reconnaître une action à partir du comportement d'un ensemble d'objets détectés dans la séquence. Dans notre graphe d'objets $\mathcal{G}_o = (\mathcal{V}, \mathcal{E})$ l'ensemble des nœuds $\mathcal{V}$ contient tous les nœuds d'objets dans la séquence.

Chaque nœud est associé à un vecteur de caractéristiques. Comme pour la séquence du squelette, nous définissons la séquence d'objets à représenter par la séquence des coordonnées 2D occupées par le centre de la boîte englobante pendant l'action. Ainsi, dans le vecteur de

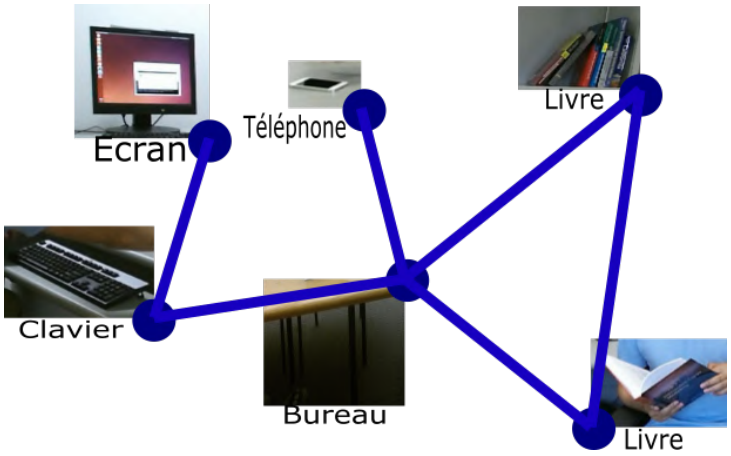


FIGURE 6.4 – Graphe-objets dont seuls les arcs spatiaux sont représentés, pour le jeu de données Watch-n-Patch office.

caractéristiques, nous ajoutons les coordonnées du centre de la boîte englobante, un indice de confiance lié au détecteur et une étiquette nature représentant la classe de l'objet.

Il existe deux types d'arêtes $\{\mathcal{E}_S, \mathcal{E}_T\} \in \mathcal{E}$ et deux nœuds $(v_1, v_2) \in \mathcal{V}$, sont reliés par un arc spatial $e \in \mathcal{E}_S$ s'ils peuvent interagir. Les mêmes objets sont reliés par un arc temporel $e_t \in \mathcal{E}_T$ entre deux images successives. Cela permet d'ajouter des connaissances a priori dans la structure du graphe grâce aux arcs spatiaux $e \in \mathcal{E}_S$ issues de notre expérience et des relations sémantiques. Par exemple, dans la plupart des cas, un couteau n'interagit pas avec une bouteille, alors qu'un verre peut interagir avec une bouteille lorsque nous y versons de l'eau. Même si divers objets sont mis en jeu dans un jeu de données, nous définissons un unique graphe-objet contenant tous les objets en interaction. Les multiples objets de même nature, par exemple la présence de trois verres est également pris en compte au préalable. Le nombre maximal d'objets de même nature présent au sein d'une même séquence est connu lors de la construction du graphe, par exemple nous considérons deux fois l'objet livre dans le jeu de données Watch-n-Patch. Un exemple de graphe-objet est présenté sur la figure 6.4, pour plus de clarté seules les arrêtes spatiales sont représentées.

6.2.4 Fusion tardive des deux réseaux

Conçus de cette manière, les deux modèles des réseaux à convolution pour graphes sont compacts quant au nombre de nœuds ainsi que de part la taille de leurs vecteurs de caractéristiques qui leur sont associés. L'utilisation mémoire et les temps d'entraînement sont donc réduits en conséquence. Il devient donc commode d'utiliser des outils d'optimisation des hyper-paramètres pour régler les paramètres d'entraînement. Étant donné une métrique à optimiser et un algorithme à exécuter, ces outils visent à trouver les paramètres optimaux. Suite à notre étude proposée en chapitre 3, nous utilisons l'outil SMAC [Hutter 2011b] pour optimiser les paramètres suivants : initialisation du taux d'apprentissage, *dropout* et taille des mini-lots (*mini-*

TABLE 6.1 – Détail de la distribution des classes au sein des jeux de données et leur nombre de clips vidéo (Nc).

Jeu de données	Nc	Distribution (% par classe)
CAD-120	1149	[23,30,3,3,3,15,4,3,1,14]
Watch-n-Patch ofc.	1148	[12,16,21,6,4,14,9,9,5,3]
Watch-n-Patch ktc.	1207	[15,11,8,8,6,10,6,17,6,6,6]
Charades	7197	[1,4,1,1,2,3,1,2,15,2,2,0,4,1,2,1, 10,1,6,3,2,2,4,10,.4,3,3,1,2,4,1,5,1]

batches). Nous choisissons d'optimiser par rapport au F1-score pour surmonter les problèmes liés aux jeux de données éventuellement déséquilibrés.

Avec les deux modèles, nous avons un modèle spécialisé pour prédire les actions où le mouvement du corps est très caractéristique et un autre modèle spécialisé pour caractériser l'évolution de la configuration des objets à partir de leurs trajectoires et de la description des interactions possibles. Nous proposons une fusion de leurs prédictions respectives. Nous avons deux vecteurs correspondant aux deux couches softmax : un pour le modèle graphe-squelette et un autre pour le modèle graphe-objet. Nous proposons une stratégie qui prend en entrée des clips vidéo qui sont d'abord pré-traités pour extraire les positions du squelette et la position objets lorsqu'ils ne sont pas fournis par le jeu de données.

Nous obtenons ainsi deux vecteurs de prédiction pour chacun des modèles qui sont ensuite concaténés. Cette concaténation est reliée à une couche dense de même taille que le nombre de classes, comme illustré sur la figure 6.1 avec seulement $N = 4$ classes. Il n'y a donc que des paramètres $N^2 + N$ à apprendre (poids N^2 liés à la couche dense et biais N liés à l'activation). Comme nous l'avons vu au chapitre 5, cette interconnexion permet de tirer profit des deux modèles dans la décision finale. Nous appelons cette approche SKLOGCN ci-après.

6.3 Évaluations

Dans cette section nous présentons d'abord le protocole expérimental, suivi d'une présentation et d'une discussion de nos résultats.

6.3.1 Protocole expérimental

Dans cette sous-section nous précisons la manière dont nous pré-traitons les séquences issues des jeux de données, les métriques utilisées pour évaluer notre proposition de réseaux de neurones à convolution pour les graphes détaillée dans la section précédente.

Jeux de données - L'approche que nous proposons combine les observations liées à la posture et aux objets dans deux réseaux à convolution compacts pour les graphes spatio-temporels. Pour évaluer cette proposition, nous choisissons des jeux de données présentant de riches interactions

entre les objets et la personne et notamment des interactions entre les objets. Un des avantages de disposer des séquences d'actions c'est qu'elles permettent d'observer les positions d'objets de même nature qui sont présents tout au long de l'exécution de différentes actions. Ainsi, nous privilégions CAD-120 [Koppula 2013c], Watch-n-Patch office, Watch-n-Patch kitchen [Wu 2015a] et Charades [Sigurdsson 2016]. Ici nous n'exploitons que des données 2D et donc nous menons également les évaluations sur Charades, ce jeu présente certains challenges supplémentaires : les séquences plus longues dans un environnement non contrôlé et des actions sont concomitantes. Les évaluations menées sur Charades sont préliminaires.

Pré-traitements des données vidéo - Les séquences vidéo des jeux de données CAD-120 et Watch-n-Patch sont découpées en clips vidéo où un clip vidéo représente une action. Lors de l'entraînement et de la phase de test, nous fixons la dimension temporelle maximale à 16 trames. Si la durée d'un clip vidéo est inférieure à 16 trames, nous mettons à zéro (*zero-pad*) les positions correspondant aux trames manquantes. Si la longueur d'un clip vidéo est supérieure à 16, nous sous-échantillonnons le clip vidéo. Afin de construire nos réseaux à convolution pour les graphes, nous avons besoin de la nature des positions 2D des articulations de la personne exécutant l'action sur la vidéo ainsi que celles des objets qui y sont mis en jeu. Les positions du squelette sont inférées par OpenPose [Cao 2017] pré-entraîné sur MSCOCO keypoint challenge [Lin 2014a]. Les détections d'objets sont soit fournies comme vérité terrain par les jeux de données comme c'est le cas avec CAD-120, soit détectées par SSD [Liu 2016b] pré-entraîné sur MSCOCO [Lin 2014b] pour Watch-n-Patch. Sur Charades les pré-traitements sont les mêmes à la différence qu'il y a dans la vérité terrain des actions concomitantes. De plus la vérité terrain contient les informations relatives à la nature des objets mais pas à leur position dans l'image, nous avons donc utilisé le détecteur SSD.

Gestion des objets Les détections d'objets sont caractérisées par les coordonnées d'une boîte englobante et d'une score de confiance associé à une classe. Dans notre réseau nous retenons le centre de la boîte englobante comme position 2D de l'objet. Lorsque les détections des objets sont fournies par la vérité terrain du jeu de données, nous y associons un score de confiance de 1. En revanche lorsque les détections émanent d'un détecteur d'objet nous y associons la classe la plus probable et son score associé. Nous avons ces deux cas de figure car tous les jeux de données ne fournissent pas les même informations dans la vérité terrain.

Métriques pour les évaluations - Il existe diverses métriques d'évaluation pour les tâches de classification multi-classe. Nous évaluons les deux modèles de graphes et leur fusion proposés dans la section 6.2 avec deux métriques. La première est la justesse, qui est définie comme suit :

$$\text{justesse} = \frac{\text{nombre de prédictions correctes}}{\text{nombre total de prédictions}} \tag{6.3}$$

Cette métrique quantifie le rapport entre les actions correctement reconnues et le nombre total d'actions à reconnaître, indépendamment du nombre d'échantillons dans chaque classe. Si une

TABLE 6.2 – Liste des hyper-paramètres optimisés, leurs intervalles de recherche et leurs valeurs initiales respectifs.

Hyper-paramètre	Intervalle	Initialisation
Initialisation du taux d'apprentissage	[0.01, 0.1]	0.1
Dropout	[0.5, 1]	0.7
Taille des mini-lots	[1, 160]	16

classe est largement sous-représentée, la justesse ne reflète pas la capacité du modèle a reconnaître cette classe en particulier. Nous comparons également notre modèle à la littérature selon le F1-score moyenné sur l'ensemble des classes. Le F1-score est une mesure qui prend en compte la précision et le rappel des différentes classes d'actions.

Pour évaluer notre approche sur le jeu de données Charades [Sigurdsson 2016] nous utilisons le code d'évaluation et les métriques indiquées par les auteurs. Puisqu'il y a des actions concomitantes, l'ensemble de la séquence vidéo y est considérée dans un problème de classification multi-label et ils utilisent la moyenne de la précision moyenne mAP (*Mean Average Precision*). Compte tenu d'un classement descendant k des vidéos pour la classe de test c, la précision moyenne (AP) par classe est définie comme suit :

$$\text{AP(c)} = \frac{\sum_{k=1}^{n} (P(k) \times rel(k))}{\text{Nombre d'échantillons dans la classe c}}, \tag{6.4}$$

où n est le nombre total de vidéos, P(k) est la précision au rank k de la liste, rel(k) est une fonction indicatrice égale à 1 si la vidéo classée au rang k est un vrai positif, et à zéro sinon.

Apprentissage du modèle Les réseaux à convolution pour les graphes sont entraînés en utilisant la fonction de perte (*loss*) classique pour des problématiques de classification multi-class : l'entropie croisée par catégories (*categorical cross entropy*). L'objectif pour le réseau est minimiser l'entropie croisée par catégories avec y la vérité terrain, $\hat{y}$ l'action prédite sur l'ensemble d'entraînement de taille M, C le nombre de classes, qui est définie comme suit :

$$CCE(\hat{y}, y) = -\frac{1}{M}\left(\sum_{i=1}^{M}\left(\sum_{j=1}^{C} y_{ij} log(\hat{y}_{ij})\right)\right). \tag{6.5}$$

L'entraînement et les évaluations sont menés sur une carte graphique NVIDIA 1080Ti.

En raison de la taille réduite des réseaux et des temps de formation, il est plus facile d'essayer différentes configurations des hyper-paramètres. Au lieu de s'appuyer sur une recherche exhaustive, nous utilisons un outil d'optimisation des hyper-paramètres issu de la littérature : SMAC [Hutter 2011b]. SMAC vise à construire un modèle de la fonction objectif afin de tester à chaque itération un ensemble de paramètres prometteurs comme nous l'avons vu au chapitre 3. La liste des paramètres du réseau que nous avons retenus pour l'optimisation ainsi que les intervalles de recherche sont décrits sur le tableau 6.2. Ce sont des paramètres classiques et largement

répandus qui sont souvent à définir lors de l'entraînement de réseaux de neurones à convolution classiques ou ici avec des convolutions appliquées aux graphes.

6.3.2 Résultats et discussion

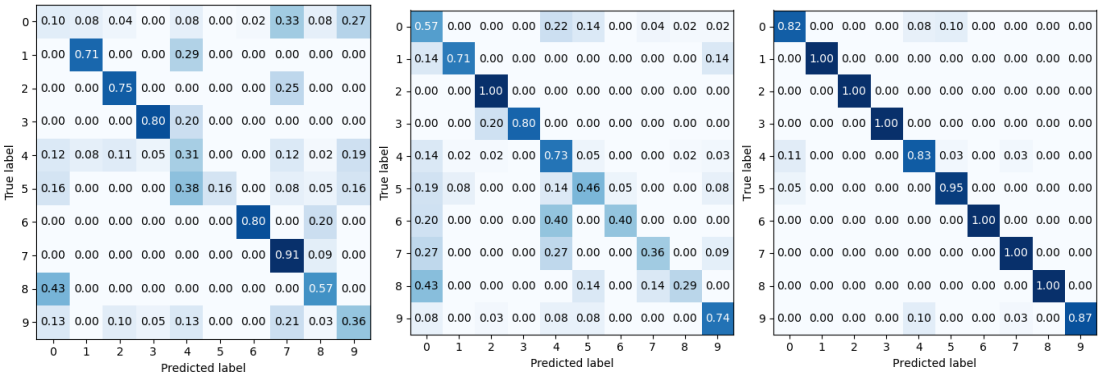
Nous présentons tout d'abord les résultats obtenus à partir de deux modèles de graphes entraînés séparément et les gains observés après leur fusion. Sur le tableau 6.3, nous pouvons voir que le modèle graphe-squelette surpasse le modèle graphe-objet lorsque nous comparons la justesse brute ou le mAP sur l'ensemble des jeux de données. En effet, un avantage du modèle graphe-objet (O) est sa capacité à discriminer certaines configurations d'objets par rapport aux actions à reconnaître. Cependant, si des actions particulières et leurs configurations d'objets associées offrent une grande spécificité, il s'agit d'actions moins fréquentes dans l'ensemble de données, et cette performance compte donc moins dans le calcul de la justesse globale. Par exemple, l'action *verser* dans le jeu de données CAD-120 implique que la bouteille de lait soit à proximité du bol et cette action ne représente que 3 % des actions comme indiqué sur le tableau 6.1. Nous étudions maintenant le gain dérivé après leur fusion, si les deux modèles ont une grande spécificité envers des classes d'actions différentes, nous observerons une meilleure précision globale. Après fusion, nous obtenons des gains en justesse de +7 points de pourcentage (pp) sur Watch-n-Patch kitchen, +4 pp sur Watch-n-Patch office et +20 pp sur CAD-120 et des gains en mAP de +8 pp pour Charades.

TABLE 6.3 – Justesse des modèles ST-GCN (S) avec le squelette seul, ST-GCN (O) avec les objets seuls et la performance de leur fusion au moyen d'une couche dense SKLOGCN. Le gain de performance est exprimé en point de pourcentage par rapport au modèle (S). Les performances sont exprimées par la justesse pour l'ensemble des jeux de données sauf pour Charades où la performance est exprimée par le mAP.

Jeu de données	**Pli**	**(S)**	**(O)**	**SKLOGCN**	**Gain vs. (S)**
Charades	R1	19,7	15,1	27,2	+8
WnP ktc.	FO	76	42	83	+7
WnP ofc.	FO	78	27	82,1	+4
CAD-120	S1	61	40	79,2	+18
	S2	66	31	86,6	+20
	S3	60	40	83,1	+23
	S4	58	34	77,4	+19
Moyenne		61	37	82	+20
CAD-120	R1	64	36	88	+24
	R2	66	34	86	+20
	R3	67	28	85	+18
	R4	69	33	86	+17
Moyenne		67	33	86	+20

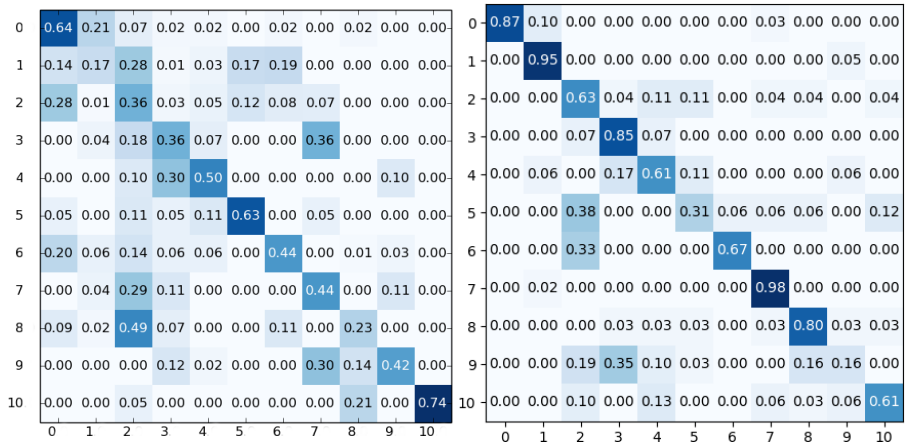
Le tableau 6.3 présente deux protocoles d'évaluation pour CAD-120. Le premier, appelé

cross-subject, (S1,...,S4) suit les instructions des auteurs du jeu de données : la validation-croisée est construite de telle sorte à apprendre le modèle sur 3 acteurs et à tester sur le 4ème acteur restant. Ce mode d'évaluation vise à évaluer si l'approche est bien robuste aux différences que nous retrouvons dans la taille des acteurs et éventuellement leur manière d'exécuter les actions suivant s'ils sont droitiers ou gauchers. Afin de nous comparer à la récente approche appelée par la suite GPNN [Qi 2018b], nous suivons également leur protocole d'expérimentation, que nous appelons cross-random, qui diffère de celui fourni par les auteurs du jeu de données CAD-120. Dans cette expérience, les différents plis sont construits de manière aléatoire avec un 80 % des données dans l'ensemble d'entraînement et 20 % dans l'ensemble de test, cf. lignes R1,...,R4.



(a) Graphe-objet (b) Graphe-squelette (c) Fusion avec couche dense.

FIGURE 6.5 – Matrices de confusion des différents réseaux et de leur fusion sur le jeu de données CAD-120, pli R1. La vérité terrain se trouve sur les lignes et les prédictions sur les colonnes.



(a) Graphe-objet (b) Graphe-squelette

FIGURE 6.6 – Matrices de confusion des différents réseaux sur le jeu de données Watch-n-Patch kitchen, pli F0. La vérité terrain se trouve sur les lignes et les prédictions sur les colonnes.

TABLE 6.4 – Comparaison des performances de notre approche avec celles de la littérature en fonction du F1-score sur le jeu de données CAD-120, suivant les deux protocoles d'évaluations.

Protocole	**Approche**	**F1-score (%)**
cross-subject	Koppula et al. [Koppula 2013c]	80,4
	S-RNN [Jain 2016a]	83,2
	Notre approche SKLOGCN	**85,1**
cross-random	GPNN [Qi 2018b]	88,9
	Notre approche SKLOGCN	**88,5**

TABLE 6.5 – Comparaison des performances de notre approche avec celles de la littérature en fonction du F1-score sur le jeu de données Watch-n-Patch.

Environnement	**Approche**	**F1-score (%)**
kitchen	**Notre approche**	**82,6**
office	PoT [Ryoo 2015]	49,9
	KMHIS [Kataoka 2016]	59,8
	ANBM [Maurice 2019]	76,1
	Notre approche SKLOGCN	**81,4**
kitchen + office	QSHPSC [Qi 2017]	71,9
	GEPHAPP [qi2]	81,5
	Notre approche SKLOGCN	**83,1**

Pour une analyse approfondie, nous fournissons sur la figure 6.5 les matrices de confusion obtenues à partir des différents modèles de graphes et de leur fusion sur CAD-120. Sur les diagonales, nous pouvons clairement voir que les performances des modèles se complètent les unes les autres en ce qui concerne certaines classes. Notons également que les sources de confusion diffèrent.

Sur Watch-n-Patch kitchen, nous pouvons voir sur la figure 6.6 que le modèle graphe-squelette détecte avec une justesse de 0,98 et 0,87 les actions *sortir-de-la-cuisine* et *attraper-depuis-le-frigo* respectivement. Alors qu'il différencie difficilement l'action *verser* des autres et de *déplacer-la-bouilloire*. Cette dernière est confondue avec *sortir-de-la-cuisine* ou *attraper-depuis-le-four*. Lorsque nous utilisons le graphe-objets, il est plus facile de lever certaines ambiguïtés car par exemple dans *sortir-de-la-cuisine* tous les objets sont statiques ce qui n'est pas le cas de l'action *attraper-depuis-le-four*, où l'ouverture de la porte du four implique un agrandissement de sa boîte englobante. Les performances et les sources de confusion sont différentes et la fusion SKLOGCN peut alors bénéficier des deux inférences.

Les auteurs de GPNN [Qi 2018b] montrent un F1-score de 88,9 pour la reconnaissance de l'action, alors que nous obtenons en moyenne 88,5 comme indiqué dans tableau 6.4. Nous comparons les résultats de notre approche à la littérature sur le jeu de données Watch-n-Patch sur le tableau 6.5. Nous observons une amélioration globale sur les deux environnements de 1,6 pp dans le F1-score par rapport à GEPHAPP [qi2], approche très récente car publiée en 2020.

TABLE 6.6 – Comparaison des performances de notre approche avec celles de la littérature en fonction du mAP sur le jeu de données Charades.

Approche	mAP
C3D [Sigurdsson 2016]	10,9
IDT [Sigurdsson 2016]	17,2
MultiScale TRN [Zhou 2018]	25,2
NLNN [Wang 2018a]	37,5
STRG [Wang 2018b]	**39,7**
Notre approche SKLOGCN	**27,2**

TABLE 6.7 – Comparaison de l'usage mémoire et de la vitesse avec différentes approches de la littérature. Le nombre de paramètres corresponds au nombre de paramètres du modèle lors de l'apprentissage. La vitesse correspond au nombre de clips vidéo traité par seconde lors de la phase de prédiction.

Approche	Nombre de paramètres	Vitesse
C3D [Tran 2015]	15 929 225	17
GPNN [Qi 2018b]	24 356 171	11
Notre approche SKLOGCN	**1 581 669**	**80**

Sur le tableau 6.6, nous comparons nos résultats préliminaires sur Charades à la littérature. Wang *et al.* [Wang 2018a, Wang 2018b] proposent d'abord une classe de réseaux neuronaux qui capturent les dépendances à longue terme via des opérations non locales et aussi avec STRG [Wang 2018b] d'utiliser des graphes et le réseau I3D [Carreira 2017]. I3D est un réseau de neurones avec 25 millions de paramètres, contrairement à notre approche avec 1,5 millions de paramètres, comme indiqué dans le tableau 6.7. Nous pouvons constater que notre approche est comparable à celle de [Zhou 2018], mais des recherches plus approfondies concernant l'intégration des caractéristiques des nœuds ou des graphes supplémentaires pourraient permettre de tirer davantage profit de notre approche légère et modulaire.

Le nombre de paramètres liés à notre modèle par rapport à C3D et GPNN [Qi 2018b] est affiché sur le tableau 6.7. Notre modélisation requiert 15 (resp. 10) fois moins de paramètres que dans la modélisation GPNN (resp. C3D). Le nombre de paramètres de notre modèle dépend du nombre de nœuds du graphique, du nombre d'images choisies pour sous-échantillonner la vidéo, de l'étendue du voisinage pendant l'opération de convolution. En ce qui concerne la vitesse de prédiction, nous sommes capables de traiter 7 fois plus de clips vidéo par seconde par rapport au GPNN et presque 5 fois par rapport au C3D.

6.3.3 Analyse complémentaire

TABLE 6.8 – Poids CAD-120 R2. Poids en rouge lorsque le graphe-squelette (S) est favorisé et en bleu lorsque le graphe-objet (O) est favorisé pour une action.

	reach	pour	eat	drink	move	place	clean	open	close	null
reach (S)	21,21	6,16	0,00	0,22	18,63	24,93	0,00	8,62	9,77	26,34
pour (S)	0,00	74,68	0,76	0,06	0,01	0,00	8,42	49,60	2,18	39,69
eat (S)	0,00	0,00	54,73	38,66	7,79	0,03	0,00	3,61	0,01	0,00
drink (S)	0,00	0,00	56,85	65,33	4,85	0,00	0,00	3,07	0,01	0,00
move (S)	13,75	0,27	0,01	0,01	23,77	28,06	0,00	8,24	0,40	27,66
place (S)	13,49	12,40	18,59	7,02	8,93	36,30	6,68	0,71	10,59	25,63
clean (S)	27,50	0,01	0,10	0,43	0,18	16,08	32,26	15,96	0,26	0,00
open (S)	10,55	0,19	0,00	0,00	35,50	11,35	13,34	17,96	2,91	0,00
close (S)	7,79	0,13	0,00	9,60	14,62	40,59	0,29	5,86	32,86	0,00
null (S)	17,18	2,18	0,00	0,02	17,71	24,16	7,65	13,22	14,53	45,22
reach (O)	33,16	5,29	7,82	0,24	22,10	5,07	0,11	8,96	15,35	5,00
pour (O)	0,36	88,06	1,37	0,46	40,46	36,70	0,09	0,10	0,12	0,00
eat (O)	28,08	0,04	40,96	4,79	13,00	0,04	0,00	10,84	3,56	0,13
drink (O)	5,20	0,02	4,91	45,30	16,11	0,04	0,00	24,25	0,03	8,51
move (O)	0,87	0,89	1,01	18,88	36,36	5,59	12,84	0,66	0,44	31,21
place (O)	0,93	0,56	10,36	8,30	9,71	35,53	2,34	7,35	0,40	20,11
clean (O)	28,59	12,71	8,00	3,32	1,16	21,60	40,51	0,06	21,64	0,00
open (O)	15,30	17,81	0,07	0,47	0,33	25,40	4,68	68,52	5,63	58,80
close (O)	6,98	0,23	1,89	0,31	11,33	24,02	9,22	0,00	24,33	34,66
null (O)	34,77	0,02	0,51	0,32	32,43	0,03	0,00	13,53	4,58	75,09

Analyse des poids de la couche dense - Les poids de la couche dense peuvent favoriser un modèle par rapport à un autre, mais ils peuvent aussi englober les inter-corrélations entre les prédictions d'actions. Les poids apprennent à saisir la spécificité de chacun des modèles. Lorsque les actions ne peuvent pas être distinguées par l'observation des objets, les poids mettent l'accent sur la prédiction du modèle squelette. Pour certaines actions telles que verser, déplacer, ouvrir, l'accent est mis sur la prédiction du modèle de l'objet comme nous pouvons le voir sur le tableau 6.8. Alors que pour les actions de manger, de boire et de placement l'accent est placé sur la prédiction du squelette.

6.4 Conclusion

Dans ce chapitre, nous proposons deux réseaux de neurones à convolution pour graphes, modulaires et légers pour la reconnaissance des actions. Un premier se base sur un graphe du squelette à l'instar de celui proposé par [Yan 2018], un second qui lui modélise les interactions entre objets. Les prédictions des deux réseaux sont fusionnées à travers une couche dense, ce qui permet d'apprendre des poids qui peuvent englober les interconnexions qui existent entre la

GT : place, prediction : place | GT : reach, prediction : reach | GT : move, prediction : move | GT : pour, prediction : pour

FIGURE 6.7 – Résultats des prédictions après fusion. En haut : sur une séquence d'activité *making cereals* de CAD-120 référence 1204173536 ainsi que la vérité terrain correspondante.

trajectoire du squelette et l'évolution de la configuration des objets dans la scène pendant une action.

Nous démontrons que les deux réseaux atteignent une grande spécificité vis-à-vis de certaines actions et qu'ils se complètent l'un l'autre. Grâce à cette complémentarité, la couche dense est capable de lever des ambiguïtés. Nous montrons sur trois jeux de données de la littérature, CAD-120, Watch-n-Patch et Charades, que l'ajout d'un réseau de graphes d'objets et notre stratégie de fusion permet des gains de performance en précision de respectivement +20 pp, +5.5 pp, +8 pp par rapport à une approche de base. Nous présentons un réseau beaucoup plus compact par rapport aux réseaux neuronaux de graphes (GNN) ou au réseau CNN 3D classique avec respectivement 15 et 10 fois moins de paramètres.

Les investigations futures portent sur la poursuite des évaluations sur Charades, l'ajout d'autres caractéristiques aux nœuds objets, par exemple la vitesse. Nous prévoyons également d'étudier la conception d'un graphe squelette-objet unique.

Ces travaux nous ont permis d'avoir un article en soumission à la conférence IEEE WACV, dont la référence complète se trouve page vii.

CHAPITRE 7

Conclusion

7.1 Synthèse des approches développées et plus-values associées

Le but de cette thèse est de proposer et d'évaluer différentes approches pour la reconnaissance d'actions grâce à la perception conjointe explicite de l'homme et des objets. Nous avons exploré plusieurs stratégies ainsi que diverses possibilités offertes par les capteurs RGB-D, à savoir l'utilisation du flux couleur seul ou combiné aux cartes de profondeur associées.

Notre première contribution, très ingénierie, consiste en l'étude de 4 outils d'optimisation des hyper-paramètres pour un algorithme de vision par ordinateur. Cette étude a été menée dans le cadre d'un projet sur la vidéo-surveillance. De part sa facilité d'utilisation, le nombre d'itérations nécessaire et la stabilité de ses résultats, un outil émerge de notre évaluation comparative : SMAC [Hutter 2011b]. Grâce à la réalisation de cette étude, les hyper-paramètres de différentes approches de reconnaissance d'actions ont ensuite été optimisés par SMAC comme indiqué dans les chapitres 46. Ainsi d'après notre retour d'expérience, l'utilisation des ces outils s'avère concluante de part les gains de performance observés quelle que soit l'approche.

Notre première stratégie de reconnaissance repose sur l'observation conjointe de la posture humaine et des objets à partir de détecteurs de la littérature : OpenPose [Cao 2017] et SSD [Liu 2016a]. Grâce au capteur de profondeur et aux données d'étalonnage caméra nous avons exprimé leurs positions dans le repère monde 3D. Ces percepts sont ensuite pris en considération dans une approche bayesienne dénommée ANBM. Nous avons créé des modèles pour chacune des actions. Cette approche est adaptée aux jeux de données qui présentent des classes d'actions déséquilibrées. L'utilisation du capteur RGB-D nous permet d'être plus robuste aux variations de points de vue caméra/scène que les approches se basant uniquement sur des descripteurs 2D. Le contexte temporel, au travers des transitions entre deux actions successives, est également pris en compte car l'exécution d'une action est souvent corrélée à la suivante. L'originalité de cette approche est la prise en compte dans un modèle unifié des postures humaines, des interactions entre objets et de cette cohérence temporelle entre actions. Cette approche est évaluée avec succès sur les jeux de données CAD-120 et Watch-n-Patch office.

La prise en compte de la cohérence temporelle dans le séquencement des actions permet de limiter certaines prédictions d'actions erronées lorsqu'elles transgressent les relations logiques observées. Forts de ce constat nous avons adapté un réseau de la littérature C3D [Tran 2015] en ajoutant une couche récurrente GRU dans le chapitre 5, ce qui en améliore les performances. Lors de l'analyse des matrices de confusion des approches ANBM et C3D-GRU, nous notons une certaine complémentarité dans les performances. Nous avons donc proposé une stratégie de fusion originale afin de tirer parti des prédictions de ce réseau C3D-GRU et des prédictions de

TABLE 7.1 – Récapitulatif de la justesse en pourcentage % des approches proposées sur différents jeux de données publics.

Approche	CAD-120	Watch-n-Patch ofc.	Watch-n-Patch ktc.
ANBM (chapitre 3)	78	76	-
C3D-GRU (chapitre 4)	66	87	-
ANBM-C3D-GRU (chapitre 4)	86	93	-
SKLOGCN (chapitre 5)	82	83	82

ANBM. La fusion tardive proposée au moyen d'une couche dense permet de favoriser soit l'une ou l'autre des approches mais également d'utiliser les inter-corrélations entre les prédictions des deux approches.

Nous proposons une nouvelle stratégie dans le chapitre 6. La contribution proposée réside dans la compacité du modèle pour la reconnaissance d'actions à partir des positions 2D de la posture humaine et des objets. Nous avons mené les évaluations sur les mêmes jeux de données que précédemment ainsi que des évaluations préliminaires sur un jeu de données supplémentaire : Charades. Charades présente un nombre plus important de classes ainsi que de nouveaux challenges. Cette approche modulaire repose sur la création de deux graphes spatio-temporels : l'un qui modélise l'évolution de la posture du squelette à l'instar de [Yan 2018] et un nouveau qui modélise les objets de la scène et les interactions entre eux. L'avantage de cette approche modulaire est tel qu'il est possible d'ajouter un réseau de convolutions à graphes supplémentaire. La complémentarité de ces deux modèles vis-à-vis de classes d'approches différentes a permis d'exhiber des gains de performance garantissant une modélisation compacte et une faible utilisation de mémoire GPU en apprentissage et en inférence.

Nous présentons ci-dessus une synthèse des performances de nos différentes approches dans le tableau 7.1. En absolu, la stratégie ANBM-C3D-GRU donne les meilleures performances sur les deux jeux de données CAD-120 et Watch-n-Patch (environnement office). Cependant il s'agit d'une approche plutôt lourde à mettre en place. Si nous sommes en présence de contraintes de temps de calcul et d'utilisation mémoire, comme c'est le cas pour des applications en embarqué, alors l'approche compacte SKLOGCN est une bonne alternative. Nos évaluations préliminaires sont prometteuses. L'ajout de contraintes temporelles entre actions successives, devrait encore améliorer les performances. Des investigations supplémentaires sont donc en cours dans ce sens.

7.2 Perspectives

Suite à ces travaux de nombreuses perspectives à court et moyen termes sont envisagées.

Perspectives à court terme

Investigations complémentaires sur l'approche SKLOGCN - Comme vu au chapitre 5 et tableau récapitulatif 7.1, l'ajout de la cohérence temporelle est un élément contextuel clé pour

augmenter les performances des systèmes de reconnaissance d'actions. En effet, de nombreuses actions ont des caractéristiques déterministes ou supposent qu'une action ait été exécutée auparavant, par exemple on ne peut pas boire un verre d'eau si le verre n'a pas été rempli au préalable. Il serait intéressant d'adapter notre approche avec réseaux de convolutions à graphes à la prise en compte de la nature de l'action précédente. Dans le chapitre 6 nous menons des évaluations préliminaires sur le jeu de données Charades. Outre un nombre de classes plus élevé, Charades présente une difficulté supplémentaire : la présence d'actions concomitantes. Ainsi, il serait nécessaire d'adapter notre réseau et son entraînement pour pouvoir faire de la prédiction multi-labels. Les résultats préliminaires prometteurs nous encouragent à poursuivre les évaluations.

Nouveaux outils d'optimisation - Suite à l'intérêt accru de la communauté Vision pour les outils d'optimisation des hyper-paramètres, de nouveaux outils ont émergé depuis la réalisation de notre étude au début de la thèse. Notamment avec la publication d'Optuna [Akiba 2019] en 2019. Cet outil a déjà permis à une approche d'apprentissage profond de détection d'objets d'atteindre la seconde place dans une compétition Kaggle [Akiba 2018]. T. Akiba*et al.* [Akiba 2019] se sont également comparés aux approches que l'on avait repérées comme pertinentes comme SMAC et TPE. En outre ils proposent une répartition des calculs qui n'était pas disponible avec SMAC. Il serait donc intéressant d'évaluer quel impact cet outil pourrait avoir sur nos performances de l'ensemble des approches développées lors de cette thèse. Deux constats motivent ces travaux : (i) il est reporté comme étant plus rapide que SMAC, et (ii) il permettrait d'optimiser les hyper-paramètres du réseau C3D.

Ajout d'éléments de contexte - L'ensemble de nos approches repose sur l'observation d'éléments contextuels implicites ou explicites qui permettent de décrire la scène. Nous avons considéré les éléments suivants : la posture 2D ou 3D, la nature et position 2D ou 3D des objets, les interactions homme-objets, les interactions objet-objet et la temporalité des actions. Nous relevons qu'il existe d'autres éléments qui peuvent être pertinents. Par exemple, la connaissance de l'environnement en analysant l'arrière plan des images car certaines actions ne sont exécutées que dans certains lieux. En plus des positions, les informations relatives aux vélocités et aux accélérations des différentes articulations peuvent être ajoutées dans les caractéristiques associées aux noeuds du réseaux de convolutions de graphes. La connaissance de l'état des objets est également intéressante et permettrait de lever certaines ambiguïtés selon si un objet est vide ou plein, allumé ou éteint, ouvert ou fermé.

Reconnaissance d'activités - Comme nous l'avons vu dans la description des jeux de données dans l'état de l'art dans le chapitre 2, les actions des jeux de données CAD-120 et Watch-n-Patch sont agencées en séquences. Ces séquences décrivent des activités de plus haut niveau dont les durées d'exécution sont plus longues que les actions. Nous pourrions tenter d'effectuer une reconnaissance d'activités à partir des résultats obtenus de l'une ou l'autre des approches

décrites dans les chapitres 456 pour la reconnaissance d'actions. Si les résultats sont prometteurs cela permet de renforcer l'intérêt dans la reconnaissance d'actions qui sont plus élémentaires.

Perspectives à moyen terme

Dans les perspectives à moyen terme nous pouvons évoquer les pistes suivantes :

Nouveau jeu de données : ToyotaSmartHome [Das 2019] - Pour la reconnaissance d'activités à domicile. Il est filmé par de multiples capteurs RGB-D ce qui permet d'évaluer la robustesse au point de vue. Ce jeu de données contient 31 classes. Particulièrement pertinent car des objets interviennent tels que : livres, bouteilles, four, tasse...

Adaptation de domaine (*domain adaptation*) images et vidéos - L'adaptation de domaine est la capacité d'appliquer un algorithme entraîné sur une ou plusieurs sources vers un domaine cible différent. Par exemple les sources et la cible peuvent être issues de différentes modalités. L'entraînement sur un jeu de données et le test sur un autre nous permettrait de voir quelles sont les actions et interactions d'objets qui seraient transposables d'un jeu à un autre. Par exemple le fait de boire un verre d'eau se retrouve dans divers jeux de données qu'ils soient vidéo ou image. Ce qui permet peut-être d'apprendre un graphe de relations entre objets à partir de jeux de données très variés. Certaines relations entre objets se retrouvent dans de nombreux jeux de données car elles font parties de notre quotidien : le verre et la bouteille, le couteau avec la fourchette, des baguettes avec un bol, etc. Ces relations sont présentes dans de nombreuses images que l'on peut collecter sur le web.

Génération de données synthétiques - Toujours dans l'optique d'augmenter la taille ou la diversité des sources d'apprentissage, nous envisageons l'utilisation de données synthétiques ; on parle alors de *data augmentation*. En effet nos approches se basent sur l'observation de la posture et les objets et il serait possible à partir d'une estimation d'en générer un nombre important de manière synthétique. Ces données synthétiques peuvent être générées à partir de données RGB ou RGB-D. En effet dans les récents travaux de Z. Li *et al.* [Li 2019], ils effectuent une estimation en 3D des mouvements d'une personne et des forces appliquées sur un objet à partir d'images RGB.

Apprentissage étudiant-prof - Ce mécanisme de diffusion des connaissances (*Knowledge-distillation*) permet d'entraîner un réseau plus petit, l'étudiant, à partir d'un réseau plus grand ou entraîné à partir de plus d'informations, le prof. Il permet d'obtenir un réseau final utilisé pour la prédiction (l'étudiant) de taille réduite ce qui a un intérêt pour les applications embarquées. Il est également possible d'induire des contraintes dans l'apprentissage au travers de ce mécanisme. À l'instar de [Wang 2019] où ils effectuent de la reconnaissance d'actions précoces, en entraînant le réseau prof avec toute la durée des séquences et le réseau étudiant est contraint à n'utiliser

que les premières trames. Nous envisageons d'ajouter des contraintes dans l'apprentissage des réseaux de convolutions.

Bibliographie

[Abu-El-Haija 2016] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan et Sudheendra Vijayanarasimhan. *Youtube-8m : A large-scale video classification benchmark*. arXiv preprint arXiv :1609.08675, 2016. (Cité en pages 17 et 72.)

[Aggarwal 2011] Jake K Aggarwal et Michael S Ryoo. *Human activity analysis : A review*. ACM Computing Surveys (CSUR), vol. 43, no. 3, pages 1–43, 2011. (Cité en page 8.)

[Akiba 2018] Takuya Akiba, Tommi Kerola, Yusuke Niitani, Toru Ogawa, Shotaro Sano et Shuji Suzuki. *Pfdet : 2nd place solution to open images challenge 2018 object detection track*. arXiv preprint arXiv :1809.00778, 2018. (Cité en page 101.)

[Akiba 2019] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta et Masanori Koyama. *Optuna : A next-generation hyperparameter optimization framework*. Dans Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 2623–2631, 2019. (Cité en pages 49 et 101.)

[Alp Güler 2018] Rıza Alp Güler, Natalia Neverova et Iasonas Kokkinos. *Densepose : Dense human pose estimation in the wild*. Dans Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 7297–7306, 2018. (Cité en page 14.)

[Andriluka] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler et Bernt Schiele. (Cité en page 9.)

[Andriluka 2014] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler et Bernt Schiele. *2d human pose estimation : New benchmark and state of the art analysis*. Dans Proceedings of the IEEE Conference on computer Vision and Pattern Recognition, pages 3686–3693, 2014. (Cité en pages 2 et 53.)

[Bergstra 2011] James S Bergstra, Rémi Bardenet, Yoshua Bengio et Balázs Kégl. *Algorithms for hyper-parameter optimization*. Dans Advances in neural information processing systems, pages 2546–2554, 2011. (Cité en pages xvii, 24, 36 et 38.)

[Bernardin 2008] Keni Bernardin et Rainer Stiefelhagen. *Evaluating multiple object tracking performance : the CLEAR MOT metrics*. EURASIP Journal on Image and Video Processing, vol. 2008, pages 1–10, 2008. (Cité en page 31.)

[BoussaïD 2013] Ilhem BoussaïD, Julien Lepagnot et Patrick Siarry. *A survey on optimization metaheuristics*. Information sciences, vol. 237, pages 82–117, 2013. (Cité en pages 24, 25 et 29.)

[Breitenstein 2010] Michael D Breitenstein, Fabian Reichlin, Bastian Leibe, Esther Koller-Meier et Luc Van Gool. *Online multiperson tracking-by-detection from a single, uncalibrated camera*. IEEE transactions on pattern analysis and machine intelligence, vol. 33, no. 9, pages 1820–1833, 2010. (Cité en page 32.)

[Cao 2017] Zhe Cao, Tomas Simon, Shih-En Wei et Yaser Sheikh. *Realtime multi-person 2d pose estimation using part affinity fields*. Dans Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 7291–7299, 2017. (Cité en pages 14, 52, 53, 61, 76, 90 et 99.)

[Carreira 2017] Joao Carreira et Andrew Zisserman. *Quo vadis, action recognition ? a new model and the kinetics dataset*. Dans proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 6299–6308, 2017. (Cité en pages xvii, 10, 12, 15 et 95.)

[Carreira 2019] Joao Carreira, Eric Noland, Chloe Hillier et Andrew Zisserman. *A short note on the kinetics-700 human action dataset*. arXiv preprint arXiv :1907.06987, 2019. (Cité en pages 17, 20 et 87.)

[Chen 2015] Chen Chen, Roozbeh Jafari et Nasser Kehtarnavaz. *UTD-MHAD : A multimodal dataset for human action recognition utilizing a depth camera and a wearable inertial sensor*. Dans 2015 IEEE International conference on image processing (ICIP), pages 168–172. IEEE, 2015. (Cité en pages 7 et 15.)

[Cho 2014] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau et Yoshua Bengio. *On the properties of neural machine translation : Encoder-decoder approaches*. arXiv preprint arXiv :1409.1259, 2014. (Cité en page 72.)

[Dai 2017] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu et Yichen Wei. *Deformable convolutional networks*. Dans Proceedings of the IEEE international conference on computer vision, pages 764–773, 2017. (Cité en page 86.)

[Das 2019] Srijan Das, Rui Dai, Michal Koperski, Luca Minciullo, Lorenzo Garattoni, Francois Bremond et Gianpiero Francesca. *Toyota smarthome : Real-world activities of daily living*. Dans Proceedings of the IEEE International Conference on Computer Vision, pages 833–842, 2019. (Cité en page 102.)

[Deng 2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li et Li Fei-Fei. *Imagenet : A large-scale hierarchical image database*. Dans 2009 IEEE conference on computer vision and pattern recognition, pages 248–255. Ieee, 2009. (Cité en page 12.)

[Domhan 2015] Tobias Domhan, Jost Tobias Springenberg et Frank Hutter. *Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves*. Dans Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015. (Cité en page 24.)

[Earley 1970] Jay Earley. *An efficient context-free parsing algorithm*. Communications of the ACM, vol. 13, no. 2, pages 94–102, 1970. (Cité en page 12.)

[Ess 2008] A. Ess, B. Leibe, K. Schindler, et L. van Gool. *A Mobile Vision System for Robust Multi-Person Tracking*. Dans IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08). IEEE Press, June 2008. (Cité en page 33.)

[Everingham 2010] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn et Andrew Zisserman. *The pascal visual object classes (voc) challenge*. International journal of computer vision, vol. 88, no. 2, pages 303–338, 2010. (Cité en page 9.)

[Fan 2016] Yin Fan, Xiangju Lu, Dian Li et Yuanliu Liu. *Video-based emotion recognition using CNN-RNN and C3D hybrid networks*. Dans Proceedings of the 18th ACM International Conference on Multimodal Interaction, pages 445–450, 2016. (Cité en page 15.)

[Feichtenhofer 2016] Christoph Feichtenhofer, Axel Pinz et Andrew Zisserman. *Convolutional two-stream network fusion for video action recognition*. Dans Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1933–1941, 2016. (Cité en pages 15 et 17.)

[Ferryman 2007] JM Ferryman. *Pets 2007 video database*. Dans Proceedings of the Tenth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance PETS, pages 1–103, 2007. (Cité en page 1.)

[Ferryman 2009] J. Ferryman et A. Shahrokni. *PETS2009 : Dataset and challenge*. Dans Twelfth IEEE Int. Workshop on Performance Evaluation of Tracking and Surveillance, pages 1–6, Dec 2009. (Cité en page 33.)

[Gibson 1977] James J Gibson. *The theory of affordances*. Hilldale, USA, vol. 1, no. 2, 1977. (Cité en page 8.)

[Gibson 1979] James J Gibson. *The theory of affordances. The ecological approach to visual perception*, 1979. (Cité en page 52.)

[Girshick 2015] Ross Girshick. *Fast r-cnn*. Dans Proceedings of the IEEE international conference on computer vision, pages 1440–1448, 2015. (Cité en page 14.)

[Haftka 2016] Raphael T Haftka, Diane Villanueva et Anirban Chaudhuri. *Parallel surrogate-assisted global optimization with expensive functions–a survey*. Structural and Multidisciplinary Optimization, vol. 54, no. 1, pages 3–13, 2016. (Cité en page 27.)

[He 2017] Kaiming He, Georgia Gkioxari, Piotr Dollár et Ross Girshick. *Mask r-cnn*. Dans Proceedings of the IEEE international conference on computer vision, pages 2961–2969, 2017. (Cité en page 14.)

[Herath 2017] Samitha Herath, Mehrtash Harandi et Fatih Porikli. *Going deeper into action recognition : A survey*. Image and vision computing, vol. 60, pages 4–21, 2017. (Cité en page 10.)

[Hu 2014] Ninghang Hu, Gwenn Englebienne, Zhongyu Lou et Ben Kröse. *Learning latent structure for activity recognition*. Dans Int. Conf. on Robotics and Automation (ICRA), 2014. (Cité en pages 10, 11, 12, 14, 63 et 64.)

[Hutter 2011a] F. Hutter, H. H. Hoos et K. Leyton-Brown. *Sequential Model-Based Optimization for General Algorithm Configuration*. Dans Proc. of LION-5, 2011. (Cité en page 61.)

[Hutter 2011b] Frank Hutter, Holger H Hoos et Kevin Leyton-Brown. *Sequential model-based optimization for general algorithm configuration*. Dans International conference on learning and intelligent optimization, pages 507–523. Springer, 2011. (Cité en pages xvii, 24, 27, 28, 36, 38, 88, 91 et 99.)

[Hutter 2014] F. Hutter, , H. Hoos et K. Leyton-Brown. *An Efficient Approach for Assessing Hyperparameter Importance*. Dans Proceedings of the 31st International Conference on International Conference on Machine Learning, volume 32 of *ICML'14*, pages I–754–I–762, 2014. (Cité en page 24.)

[Ikizler 2007] Nazli Ikizler et David Forsyth. *Searching video for complex activities with finite state models*. Dans 2007 IEEE Conference on Computer Vision and Pattern Recognition, pages 1–8. IEEE, 2007. (Cité en page 10.)

[Insafutdinov 2016] Eldar Insafutdinov, Leonid Pishchulin, Bjoern Andres, Mykhaylo Andriluka et Bernt Schiele. *Deepercut : A deeper, stronger, and faster multi-person pose estimation model*. Dans European Conference on Computer Vision, pages 34–50. Springer, 2016. (Cité en pages 14 et 53.)

[Ionescu 2014] Catalin Ionescu, Dragos Papava, Vlad Olaru et Cristian Sminchisescu. *Human3.6M : Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments*. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2014. (Cité en pages 17, 19, 54 et 55.)

[Iqbal 2016] Umar Iqbal et Juergen Gall. *Multi-person pose estimation with local joint-to-person associations*. Dans European Conference on Computer Vision, pages 627–642. Springer, 2016. (Cité en page 53.)

[Jain 2016a] Ashesh Jain, Amir R Zamir, Silvio Savarese et Ashutosh Saxena. *Structural-rnn : Deep learning on spatio-temporal graphs*. Dans Proceedings of the ieee conference on computer vision and pattern recognition, pages 5308–5317, 2016. (Cité en pages 10, 12, 14 et 94.)

[Jain 2016b] Ashesh Jain, Amir R. Zamir, Silvio Savarese et Ashutosh Saxena. *Structural-RNN : Deep Learning on Spatio-Temporal Graphs*. Conference on Computer Vision and Pattern Recognition (CVPR), 2016. (Cité en pages 63 et 64.)

[Karpathy 2014] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar et Li Fei-Fei. *Large-scale video classification with convolutional neural networks*. Dans Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pages 1725–1732, 2014. (Cité en pages 17 et 72.)

[Kataoka 2016] Hirokatsu Kataoka, Yudai Miyashita, Masaki Hayashi, Kenji Iwata et Yutaka Satoh. *Recognition of Transitional Action for Short-Term Action Prediction using Discriminative Temporal CNN Feature.* Dans BMVC, 2016. (Cité en pages 15, 16, 65 et 94.)

[Kay 2017] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev*et al. The kinetics human action video dataset*. arXiv preprint arXiv :1705.06950, 2017. (Cité en page 83.)

[Kazakos 2019] Evangelos Kazakos, Arsha Nagrani, Andrew Zisserman et Dima Damen. *Epic-fusion : Audio-visual temporal binding for egocentric action recognition*. Dans Proceedings of the IEEE International Conference on Computer Vision, pages 5492–5501, 2019. (Cité en page 15.)

[Kim 2006] Kyungnam Kim et Larry S Davis. *Multi-camera tracking and segmentation of occluded people on ground plane using search-guided particle filtering*. Dans European Conference on Computer Vision, pages 98–109. Springer, 2006. (Cité en page 30.)

[Kipf 2017] Thomas N Kipf et Max Welling. *Semi-supervised classification with graph convolutional networks*. In ICLR, 2017. (Cité en pages 10, 84 et 85.)

[Kirkpatrick 1983] Scott Kirkpatrick, C Daniel Gelatt et Mario P Vecchi. *Optimization by simulated annealing*. science, vol. 220, no. 4598, pages 671–680, 1983. (Cité en page 26.)

[Koppula 2013a] Hema Koppula et Ashutosh Saxena. *Learning spatio-temporal structure from rgb-d videos for human activity detection and anticipation*. Dans International Conference on Machine Learning, 2013. (Cité en pages 11, 12, 14 et 65.)

[Koppula 2013b] Hema Swetha Koppula, Rudhir Gupta et Ashutosh Saxena. *Learning human activities and object affordances from rgb-d videos*. The International Journal of Robotics Research, vol. 32, no. 8, pages 951–970, 2013. (Cité en pages xv, 10, 16, 17, 19, 62, 63, 64, 65, 66 et 69.)

[Koppula 2013c] Hema Swetha Koppula, Rudhir Gupta et Ashutosh Saxena. *Learning human activities and object affordances from rgb-d videos*. The International Journal of Robotics Research, vol. 32, no. 8, pages 951–970, 2013. (Cité en pages 90 et 94.)

[Kuehne 2011] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio et Thomas Serre. *HMDB : a large video database for human motion recognition*. Dans 2011 International Conference on Computer Vision, pages 2556–2563. IEEE, 2011. (Cité en page 17.)

[lad 2020] *Vidéosurveillance : Toulouse en tête des villes qui ont le plus augmenté le nombre de caméras*. paru le 20 février 2020 sur le site du journal La Dépêche, 2020. (Cité en page 1.)

[Li 2010] W. Li, Z. Zhang et Z. Liu. *Action recognition based on a bag of 3D points*. Dans 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops, 2010. (Cité en pages 17, 18 et 69.)

[Li 2019] Zongmian Li, Jiri Sedlar, Justin Carpentier, Ivan Laptev, Nicolas Mansard et Josef Sivic. *Estimating 3d motion and forces of person-object interactions from monocular video*. Dans Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 8640–8649, 2019. (Cité en page 102.)

[Lien 2020] Dylan Josh Domingo Lopez ; Cheng-Chang Lien. *CHU Surveillance Violence Detection Dataset*, 2020. (Cité en page 2.)

[Lin 2014a] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár et C Lawrence Zitnick. *Microsoft coco : Common objects in context*. Dans European conference on computer vision, pages 740–755. Springer, 2014. (Cité en pages 2, 61 et 90.)

[Lin 2014b] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár et C Lawrence Zitnick. *Microsoft coco : Common objects in*

context. Dans European conference on computer vision, pages 740–755. Springer, 2014. (Cité en page 90.)

[Lin 2017] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan et Serge Belongie. *Feature pyramid networks for object detection*. Dans Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2117–2125, 2017. (Cité en page 14.)

[Liu 2016a] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu et Alexander C Berg. *Ssd : Single shot multibox detector*. Dans European conference on computer vision, pages 21–37. Springer, 2016. (Cité en pages 14, 52 et 99.)

[Liu 2016b] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu et Alexander C Berg. *Ssd : Single shot multibox detector*. Dans European conference on computer vision. Springer, 2016. (Cité en pages 61 et 90.)

[Liu 2019] Jun Liu, Amir Shahroudy, Mauricio Perez, Gang Wang, Ling-Yu Duan et Alex C. Kot. *NTU RGB+D 120 : A Large-Scale Benchmark for 3D Human Activity Understanding*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2019. (Cité en pages 17 et 20.)

[Luo 2016] Gang Luo. *A review of automatic selection methods for machine learning algorithms and hyper-parameter values*. Network Modeling Analysis in Health Informatics and Bioinformatics, vol. 5, no. 1, page 18, 2016. (Cité en page 24.)

[Martin 2019] P. Martin, J. Benois-Pineau et R. Péteri. *Fine-Grained Action Detection and Classification in Table Tennis with Siamese Spatio-Temporal Convolutional Neural Network*. Dans 2019 IEEE International Conference on Image Processing (ICIP), pages 3027–3028, 2019. (Cité en page 1.)

[Maurice 2019] C. Maurice, F. Madrigal, A. Monin et F. Lerasle. *A New Bayesian Modeling for 3D Human-Object Action Recognition*. Dans 2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), pages 1–8, 2019. (Cité en pages 69, 70, 81 et 94.)

[Metropolis 1953] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller et Edward Teller. *Equation of state calculations by fast computing machines*. The journal of chemical physics, vol. 21, no. 6, pages 1087–1092, 1953. (Cité en page 25.)

[Minton 1992] Steve Minton, Mark Johnston, Andrew Philips et Phil Laird. *Minimizing conflicts : A heuristic repair method for constraint-satisfaction and scheduling problems*. 1992. (Cité en page 25.)

[Niepert 2016] Mathias Niepert, Mohamed Ahmed et Konstantin Kutzkov. *Learning convolutional neural networks for graphs*. Dans International conference on machine learning, pages 2014–2023, 2016. (Cité en page 86.)

[Ofli 2013] Ferda Ofli, Rizwan Chaudhry, Gregorij Kurillo, René Vidal et Ruzena Bajcsy. *Berkeley mhad : A comprehensive multimodal human action database*. Dans 2013 IEEE Workshop on Applications of Computer Vision (WACV). IEEE, 2013. (Cité en pages 17 et 54.)

[Oquab 2014] Maxime Oquab, Leon Bottou, Ivan Laptev et Josef Sivic. *Learning and transferring mid-level image representations using convolutional neural networks*. Dans Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1717–1724, 2014. (Cité en pages xvii, 9 et 13.)

[Peshkin 1999] Michael Peshkin et J Edward Colgate. *Cobots*. Industrial Robot : An International Journal, 1999. (Cité en page 1.)

[Pishchulin 2016] Leonid Pishchulin, Eldar Insafutdinov, Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, Peter V Gehler et Bernt Schiele. *Deepcut : Joint subset partition and labeling for multi person pose estimation*. Dans Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4929–4937, 2016. (Cité en page 53.)

[qi2] (Cité en pages 19, 81 et 94.)

[Qi 2017] Siyuan Qi, Siyuan Huang, Ping Wei et Song-Chun Zhu. *Predicting human activities using stochastic grammar*. Dans Int. Conference on Computer Vision (ICCV), IEEE, 2017. (Cité en pages 11, 12, 14, 63, 64 et 94.)

[Qi 2018a] Siyuan Qi, Wenguan Wang, Baoxiong Jia, Jianbing Shen et Song-Chun Zhu. *Learning human-object interactions by graph graphparsing neural networks*. Dans ECCV, 2018. (Cité en pages 63, 64, 80 et 81.)

[Qi 2018b] Siyuan Qi, Wenguan Wang, Baoxiong Jia, Jianbing Shen et Song-Chun Zhu. *Learning human-object interactions by graph parsing neural networks*. Dans Proceedings of the European Conference on Computer Vision (ECCV), pages 401–417, 2018. (Cité en pages xvii, 13, 14, 16, 19, 85, 93, 94 et 95.)

[Redmon 2018] Joseph Redmon et Ali Farhadi. *Yolov3 : An incremental improvement*. arXiv preprint arXiv :1804.02767, 2018. (Cité en page 14.)

[Ryoo 2015] Michael S Ryoo, Brandon Rothrock et Larry Matthies. *Pooled motion features for first-person videos*. Dans CVPR, 2015. (Cité en pages 16, 65, 81 et 94.)

[Sarafianos 2016] Nikolaos Sarafianos, Bogdan Boteanu, Bogdan Ionescu et Ioannis A Kakadiaris. *3d human pose estimation : A review of the literature and analysis of covariates*. Computer Vision and Image Understanding, vol. 152, pages 1–20, 2016. (Cité en page 53.)

[Shahriari 2015] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams et Nando De Freitas. *Taking the human out of the loop : A review of Bayesian optimization*. Proceedings of the IEEE, vol. 104, no. 1, pages 148–175, 2015. (Cité en pages 24 et 27.)

[Shahroudy 2016] Amir Shahroudy, Jun Liu, Tian-Tsong Ng et Gang Wang. *NTU RGB+ D : A large scale dataset for 3D human activity analysis*. Dans Proceedings of the IEEE conference on computer vision and pattern recognition, 2016. (Cité en pages 17 et 87.)

[Sharma 2013] Gaurav Sharma, Frédéric Jurie et Cordelia Schmid. *Expanded parts model for human attribute and action recognition in still images*. Dans proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 652–659, 2013. (Cité en page 9.)

[Sigurdsson 2016] Gunnar A Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev et Abhinav Gupta. *Hollywood in homes : Crowdsourcing data collection for activity understanding*. Dans European Conference on Computer Vision, pages 510–526. Springer, 2016. (Cité en pages 17, 20, 90, 91 et 95.)

[Smeulders 2013] Arnold WM Smeulders, Dung M Chu, Rita Cucchiara, Simone Calderara, Afshin Dehghan et Mubarak Shah. *Visual tracking : An experimental survey*. IEEE transactions on pattern analysis and machine intelligence, vol. 36, no. 7, pages 1442–1468, 2013. (Cité en page 30.)

[Snoek 2012] Jasper Snoek, Hugo Larochelle et Ryan P Adams. *Practical bayesian optimization of machine learning algorithms*. Dans Advances in neural information processing systems, pages 2951–2959, 2012. (Cité en pages xvii, 24, 27, 29, 36 et 38.)

[Snoek 2015] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat et Ryan Adams. *Scalable bayesian optimization using deep neural networks*. Dans International conference on machine learning, pages 2171–2180, 2015. (Cité en page 27.)

[Soomro 2012a] Khurram Soomro, Amir Roshan Zamir et Mubarak Shah. *UCF101 : A dataset of 101 human actions classes from videos in the wild*. arXiv preprint arXiv :1212.0402, 2012. (Cité en pages 17 et 19.)

[Soomro 2012b] Khurram Soomro, Amir Roshan Zamir et Mubarak Shah. *UCF101 : A dataset of 101 human actions classes from videos in the wild*. arXiv preprint arXiv :1212.0402, 2012. (Cité en page 83.)

[Toshev 2014] Alexander Toshev et Christian Szegedy. *Deeppose : Human pose estimation via deep neural networks*. Dans Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1653–1660, 2014. (Cité en page 14.)

[Tran 2015] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani et Manohar Paluri. *Learning spatiotemporal features with 3d convolutional networks*. Dans Proceedings of the IEEE international conference on computer vision, pages 4489–4497, 2015. (Cité en pages 10, 12, 70, 71, 72, 76, 83, 95 et 99.)

[Turaga 2008] Pavan Turaga, Rama Chellappa, Venkatramana S Subrahmanian et Octavian Udrea. *Machine recognition of human activities : A survey*. IEEE Transactions on Circuits and Systems for Video technology, vol. 18, no. 11, pages 1473–1488, 2008. (Cité en page 8.)

[Wang 2013a] Heng Wang et Cordelia Schmid. *Action recognition with improved trajectories*. Dans Proceedings of the IEEE international conference on computer vision, pages 3551–3558, 2013. (Cité en page 10.)

[Wang 2013b] Ziyu Wang, Masrour Zoghi, Frank Hutter, David Matheson et Nando De Freitas. *Bayesian optimization in high dimensions via random embeddings*. Dans Twenty-Third International Joint Conference on Artificial Intelligence, 2013. (Cité en page 24.)

[Wang 2018a] Xiaolong Wang, Ross Girshick, Abhinav Gupta et Kaiming He. *Non-local neural networks*. Dans CVPR, pages 7794–7803, 2018. (Cité en page 95.)

[Wang 2018b] Xiaolong Wang et Abhinav Gupta. *Videos as space-time region graphs*. Dans ECCV, 2018. (Cité en page 95.)

[Wang 2019] Xionghui Wang, Jian-Fang Hu, Jian-Huang Lai, Jianguo Zhang et Wei-Shi Zheng. *Progressive teacher-student learning for early action prediction*. Dans Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3556–3565, 2019. (Cité en page 102.)

[Watada 2010] Junzo Watada, Zalili Musa, Lakhmi C Jain et John Fulcher. *Human tracking : A state-of-art survey*. Dans International Conference on Knowledge-Based and Intelligent Information and Engineering Systems, pages 454–463. Springer, 2010. (Cité en page 30.)

[Wu 2015a] Chenxia Wu, Jiemi Zhang, Silvio Savarese et Ashutosh Saxena. *Watch-n-patch : Unsupervised understanding of actions and relations*. Dans Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4362–4370, 2015. (Cité en pages 17 et 90.)

[Wu 2015b] Chenxia Wu, Jiemi Zhang, Silvio Savarese et Ashutosh Saxena. *Watch-n-patch : Unsupervised understanding of actions and relations*. Dans Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015. (Cité en pages 62, 65, 74 et 77.)

[Wu 2016] C. Wu, J. Zhang, B. Selman, S. Savarese et A. Saxena. *Watch-Bot : Unsupervised learning for reminding humans of forgotten actions*. Dans 2016 IEEE International Conference on Robotics and Automation (ICRA), 2016. (Cité en pages 65 et 81.)

[Wu 2020] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang et S Yu Philip. *A comprehensive survey on graph neural networks*. IEEE Transactions on Neural Networks and Learning Systems, 2020. (Cité en page 86.)

[Yan 2018] Sijie Yan, Yuanjun Xiong et Dahua Lin. *Spatial temporal graph convolutional networks for skeleton-based action recognition*. Dans Thirty-second AAAI conference on artificial intelligence, 2018. (Cité en pages 13, 14, 85, 86, 87, 96 et 100.)

[Yao 2012] Bangpeng Yao et Li Fei-Fei. *Recognizing human-object interactions in still images by modeling the mutual context of objects and human poses*. IEEE transactions on pattern analysis and machine intelligence, vol. 34, no. 9, pages 1691–1703, 2012. (Cité en page 9.)

[Yao 2019] Guangle Yao, Tao Lei et Jiandan Zhong. *A review of Convolutional-Neural-Network-based action recognition*. Pattern Recognition Letters, vol. 118, pages 14–22, 2019. (Cité en page 10.)

[you 2020] *Statistiques d'utilisation de YouTube*. `https://www.youtube.com/about/press/`, 2020. [En ligne ; 07 octobre 2020]. (Cité en page 1.)

[Zhou 2016] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva et Antonio Torralba. *Learning deep features for discriminative localization*. Dans Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2921–2929, 2016. (Cité en page 9.)

[Zhou 2018] Bolei Zhou, Alex Andonian, Aude Oliva et Antonio Torralba. *Temporal relational reasoning in videos*. Dans ECCV, 2018. (Cité en page 95.)