Université de Lille

École doctorale Science de l'ingénierie et des systèmes (ENGSYS ED 632)

Thèse de doctorat

présentée et soutenue le 22 Novembre 2022

par

Rémi DUFOUR

En vue de l'obtention du grade de docteur de l'Université de Lille

Spécialité : Micro-nanosystèmes et capteurs

Segmentation d'instance dans des images fisheye et détection de points clés de squelette dans des vidéos : application à la vidéoprotection à bord du futur train autonome

Composition du jury :

Rapporteurs	
Catherine Achard	Professeure des Universités, Sorbonne Université, Paris
Samia Ainouz	Professeure des Universités, INSA Rouen Normandie, Rouen
Examinateurs	
François Brémond	Directeur de Recherche, INRIA, Sophia-Antipolis, (Président du jury)
Marion Berbineau	Directrice de Recherche, Université Gustave Eiffel, Lille, (Co-Directrice de thèse)
Olivier Lézoray	Professeur des Universités, Université de Caen Normandie, Caen, (Directeur de thèse)
Cyril Meurie	Chargé de Recherche, Université Gustave Eiffel, Lille, (Encadrant de thèse)
Invité	
Ankur Mahtani	Ingénieur R&D, IRT Railenium, Valenciennes, (Co-Encadrant de thèse)

Remerciements

Cette thèse a bénéficié de l'aide de nombreuses personnes que je tiens à remercier.

Je remercie tout d'abord ma directrice de thèse Marion Berbineau, mon co-directeur de thèse Olivier Lézoray, et mon encadrant Cyril Meurie pour leurs conseils délivrés tout au long de ma thèse.

Je remercie en particulier Cyril Meurie, qui a encadré mon stage de fin d'études d'ingénieur, m'a ensuite recruté en tant qu'ingénieur d'études, puis m'a fait part de l'opportunité d'effectuer cette présente thèse. De plus, il a été d'une aide précieuse en encadrant ma thèse, pendant laquelle il m'a transmis sa gentillesse et ses compétences.

Je dois également beaucoup à Olivier Lézoray, pour ses conseils, sa capacité d'écoute et la compréhension fine des enjeux de ma thèse.

Je tiens également à remercier mes collègues de Railenium, Ankur Mahtani et Sébastien Lefebvre, qui m'ont respectivement encadré techniquement et aidé administrativement.

Un grand merci aux autres thésards de Railenium, pour les discussions passionnées sur nos sujets de thèses.

Je remercie aussi François Brémond, président de jury pour l'intérêt qu'il a porté à l'examen de mes travaux, ainsi que Catherine Achard et Samia Ainouz pour avoir rapporté ma thèse et pour leurs retours précieux.

Ma famille m'a beaucoup apporté, en particulier pendant la dernière année de thèse, et je souhaite également la remercier ici.

Enfin, je remercie l'IRT Railenium pour tout ce qu'il m'a apporté durant ce travail. Cette thèse a été cofinancée par l'Union européenne avec le Fonds européen de développement régional que je remercie aussi ici.

FR - Résumé

Les projets de trains autonomes se multiplient à travers le monde. En France un consortium dirigé par l'IRT Railenium a pour objectif de construire un prototype de train autonome atteignant GoA4 (Grade of Automation 4) et qui serait capable de circuler sans conducteur et sans personnel humain à bord. En l'absence de personnel, les besoins de services et de sécurité des usagers doivent être pris en charge par des systèmes automatisés. De tels systèmes doivent disposer d'informations variées et détaillées, en particulier sur l'état et les actions des usagers présents à bord. Les algorithmes de vision par ordinateur, en particulier ceux basés sur l'apprentissage automatique par réseaux de neurones profonds, aussi appelés Deep Learning, ont récemment atteint des niveaux de performances convenables pour analyser des flux vidéos de caméras de surveillance. Plusieurs défis spécifiques au contexte du train autonome doivent cependant être relevés. Certaines caméras chargées d'assurer la sécurité des passagers à l'intérieur du train seront de type grand angle ou fisheye. Ces dernières produisent des images présentant des distorsions en barillet importantes, qui ne sont pas présentes dans les principales bases de données d'entraînement de la littérature, et qui permettent l'apprentissage des réseaux de neurones convolutifs modernes. Dans la littérature, une méthode a été développée pour entraîner des algorithmes de segmentation sémantique sur des images fisheye artificielles. Dans cette thèse, nous appliquons pour la première fois, cette méthode à la tâche de segmentation d'instance, et nous étudions d'une part ses performances sur deux nouvelles bases d'images réelles présentant des distorsions en barillet, mais aussi l'effet de l'initialisation et de certains paramètres. De plus, les nouveaux algorithmes de suivi de pose ont maintenant atteint une certaine maturité. Cependant, ils sont généralement top-down, et ne disposent pas d'une mémoire à long terme. Nous proposons donc dans un second temps une nouvelle méthode de détection de points clés de squelettes de personnes dans des vidéos, en adaptant un algorithme récent de Video Object Segmentation (VOS), qui dispose d'une mémoire à long terme. Ces propositions d'algorithmes ont été évaluées et validées sur des données acquises dans un modèle de train Regio2N spécifique au contexte du futur train autonome.

EN - Abstract

Autonomous train projects are multiplying around the world. In France a consortium directed by IRT Railenium has the goal to build a train prototype that achieves GoA4 (Grade of Automation 4) that would be able to navigate without a driver and without on-board staff. In the absence of staff, the needs of services and security of the passengers must be taken care of by automatic systems. Such systems must have varied and detailed information, in particular on the state and actions of the passengers on board. Computer vision algorithms, in particular those based on machine larning by deep neural networks, also called "Deep Learning" algorithms, have recently achieved a level of performance adequate to analyse video streams from surveillance cameras. However, many challenges specific to the context of autonommous trains need to be adressed. Some cameras tasked with ensuring the security of passengers inside the train will be of the wide-angle or Fisheye type. Those cameras produce images that contain important barrel distortions, that are not present in the main datasets from the literature that permit the training of modern convolutional neural networks. In the literature, a method was developed to train semantic segmentation algorithm on artificial fisheye images. In this thesis, we apply for the first time this method for the task of instance segmentation, and we study, on the one hand, its performance on two new annotated real images datasets with barrel distortions, but also the effect of initialization and certain parameters. Moreover, recent pose tracking algorithms have now achieved some degree of maturity. However, they are generally top-down, and do not make use of a long term memory. We propose a new method for person skeleton key-points detection in video, that adapts a recent Video Object Segmentation (VOS) algorithm, that makes use of a long term memory. These algorithms propositions were evaluated and validated on data acquired in a train model Regio2N specific to the context of the future autonomous train.

Table des matières

Chapitre 1

Introduction

Améliorer les services et la sécurité des usagers dans les transports publics est important pour leur popularité dans le futur. En particulier, des projets de trains autonomes sont aujourd'hui en cours de réalisation dans plusieurs pays du monde, y compris en France. Le programme Train Autonome Service Voyageurs (TASV), concerné par cette présente thèse, préparée à l'Institut de Recherche Technologique (IRT) Railenium, bénéficierait avantageusement de l'usage de caméras de sécurité capables d'analyser automatiquement l'intérieur du train. Récemment, l'analyse de flux vidéo en temps réel est devenue envisageable, grâce aux avancées dans le domaine de l'apprentissage profond, et en particulier dans deux tâches majeures de l'analyse de scène que sont : la segmentation d'instance, et le suivi de pose.

Cependant, de nombreux défis restent à relever. Tout d'abord, d'un point de vue capteur, des caméras à objectif fisheye seront utilisées pour le projet TASV. Malheureusement, les bases de données annotées publiques d'images fisheye sont aujourd'hui beaucoup plus rares et moins consistantes en taille que les bases de données d'images rectilinéaires. Hors, des bases de données de grande taille sont indispensables pour l'entraînement des algorithmes d'apprentissage profond. L'annotation d'une nouvelle base de données fisheye représente un coût et un temps considérable, c'est pourquoi une méthode permettant de s'en passer serait d'une grande utilité. Plusieurs méthodes ont été proposées dans la littérature, mais la plus efficiente d'entre elles, selon nous, utilise des images à effet fisheye synthétiques, créées à partir d'images rectilinéaires, pour entraîner par affinage un algorithme d'apprentissage profond pour la segmentation sémantique. Dans le cadre du programme TASV, il serait pertinent d'appliquer cette méthode à la tâche de segmentation d'instance, tout en essayant de conserver de bonnes performances sur les images rectilinéaires, puisque le futur train autonome utilisera les deux types de caméras.

Une deuxième tâche importante pour l'analyse vidéo de l'intérieur du train est la reconnaissance d'actions. La littérature offre un panel de méthodes de reconnaissance d'actions dans des vidéos à partir de squelettes de personnes. Pour appliquer de tels al-

gorithmes, il est nécessaire de disposer de squelettes de personnes suivis dans le temps. Obtenir ces squelettes revient à considérer le "Suivi de pose", pour lequel de nombreux algorithmes ont été proposés. Cependant, la plupart des algorithmes proposés sont top-down, c'est-à-dire, qu'ils utilisent tout d'abord un algorithme de détection de personnes, produisant des boîtes englobantes de personnes dans l'image, avant d'utiliser un algorithme d'estimation de pose pour chaque boîte englobante détectée. Mais ces algorithmes ont un défaut crucial pour une utilisation en exploitation : leur rapidité d'exécution dépend du nombre de personnes présentes dans l'image. Dans un train régional (Regio2N comme celui utilisé dans le programme TASV), le nombre de personnes à bord peut facilement atteindre plusieurs dizaines, ce qui est bien au dessus du nombre de personnes généralement présent dans les séquences vidéos des bases de données de référence en suivi de pose. À contrario, les algorithmes bottom-up détectent d'abord les parties du corps individuellement, avant de les assembler en squelettes. Cette différence rend leur rapidité d'exécution (pour l'essentiel) indépendante du nombre de personnes présentes dans l'image. Cette propriété est cruciale et rend par conséquent intéressant tout algorithme de suivi de pose bottom-up.

Dans cette thèse, nous cherchons à développer de nouveaux algorithmes d'apprentissage profond capables de traiter les défis susmentionnés. Nos contributions sont résumées ainsi :

- Une proposition de méthode capable d'améliorer les performances d'un algorithme de segmentation d'instance sur les images fisheye grâce à l'usage d'images FE (Fisheye Effect) synthétiques, tout en gardant de bonnes performances sur des images rectilinéaires.
- Une proposition de nouvel algorithme bottom-up de détection de points clés de squelettes de personnes dans des vidéos, inspiré d'un algorithme existant de VOS (Video Object Segmentation) appelé STM (Space-Time Memory), qui vise à être performant au moyen d'une mémoire de frames données en entrée plus importante qu'avec les algorithmes existants comparables.
- L'évaluation de nos deux propositions à savoir l'algorithme de segmentation d'instance et l'algorithme de détection de points clés de squelettes de personnes dans des vidéos sur de nouvelles bases de données réelles de train autonome, acquises et annotées pour les besoins de cette thèse et du programme TASV.

Ce manuscrit est organisé de la façon suivante :

- Le chapitre 2 présente une rapide vue d'ensemble des domaines de l'apprentissage automatique et de l'apprentissage profond, ainsi que des notions de bases pour aborder sereinement la compréhension des chapitres suivants.

- Le chapitre 3 présente notre méthode d'entraînement par affinage qui utilise une augmentation de données basée sur la création d'images à effet fisheye (FE) synthétiques. Cette méthode a été développée dans le but d'obtenir de bonnes performances à la fois sur des images fisheye mais aussi sur des images rectilinéaires.
- Le chapitre 4 présente notre algorithme bottom-up de détection de points clés de squelettes de personnes dans des vidéos inspiré de l'algorithme STM utilisé en VOS. Plusieurs configurations d'entraînement avec augmentation de données sont proposées et comparées dans une étude ablative.
- Le chapitre 5, dédié au contexte applicatif du futur train autonome, évalue les algorithmes proposés dans les deux chapitres précédents sur deux bases de données réelles acquises dans le cadre du programme TASV et de cette thèse préparée à l'IRT Railenium. La première base de données contient des images fisheye annotées en segmentation d'instance de personnes, tandis que la seconde contient des images rectilinéaires annotées pour la détection de points clés de squelettes de personnes.
- Le chapitre 6 conclut sur le travail présenté dans ce mémoire et propose quelques perspectives à ces travaux.

Chapitre 2

Apprentissage automatique

Dans ce chapitre, nous allons introduire des notions fondamentales pour aborder sereinement la compréhension des différents chapitres de ce manuscrit. Nous commencerons par donner une vue d'ensemble du domaine de l'intelligence artificielle. Ensuite, nous détaillerons plus spécifiquement les sous-domaines de l'apprentissage automatique et de l'apprentissage profond, en expliquant les principes et algorithmes d'optimisation qui sont au coeur du succès des techniques d'apprentissage profond aujourd'hui. La figure 2.1 illustre l'articulation entre ces trois grands domaines de recherche.

2.1 Apprentissage automatique

L'apprentissage automatique, ou Machine Learning (ML) en anglais, est un champ de recherche qui se consacre à l'étude de machines capables de s'améliorer par l'expérience ou grâce à l'exploitation de données. Le concept est mentionné pour la première fois dans [Samuel, 1959] en 1959 dans le cadre de l'apprentissage du jeu de dames. Les techniques issues de ce domaine ont été principalement appliquées à l'apprentissage de reconnaissance de formes. Une définition a été donnée en 1997 Par Tom M. Mitchell : "On dit d'un programme informatique qu'il apprend de l'expérience E relativement à

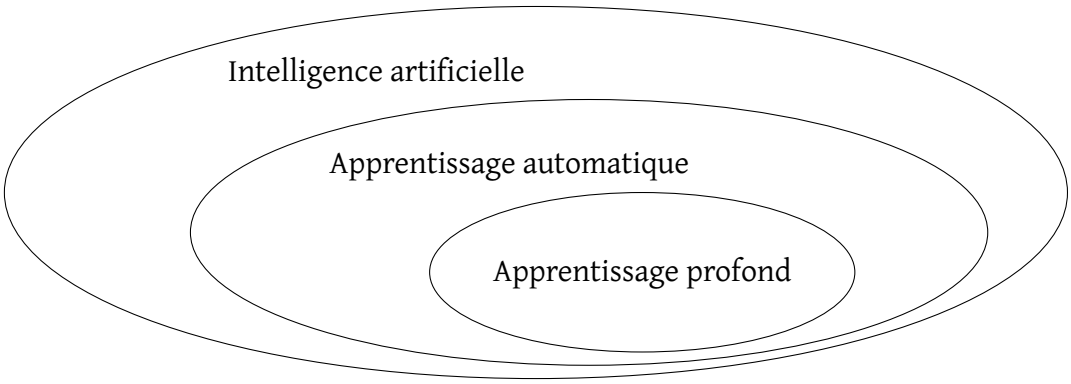


FIGURE 2.1 – Intelligence artificielle, apprentissage automatique et apprentissage profond.

un ensemble de tâches T et des mesures de performance P si ses performances sur les tâches T, mesurées par P, s'améliorent avec l'expérience E.". La croissance, au cours des dernières décennies, de la puissance de calcul disponible, de la capacité de stockage, du développement d'internet, et par conséquence, des données disponibles, ont permis un développement massif de l'apprentissage automatique et son application à de multiples domaines : traitement d'images, reconnaissance de paroles, traitement de texte (Natural Language Processing, NLP)... La plupart des approches modernes utilisent une approche connectionniste basée sur les réseaux de neurones profonds. Aujourd'hui, les hardwares (GPU Nvidia) et plusieurs bibliothèques (Tensorflow, Pytorch, Caffe, JAX) sont disponibles et facilitent la tâche de conception de nouveaux algorithmes d'apprentissage automatique.

Dans ce chapitre, nous allons présenter quelques uns des algorithmes d'apprentissage automatique qui ont précédé la popularisation de l'apprentissage profond durant les dernières décennies, afin de disposer d'une vue d'ensemble du domaine et donc de mieux comprendre les raisons d'être de l'apprentissage profond. Nous allons également voir les principes de base du perceptron multi-couches et de l'entraînement de tels perceptron au travers de la rétropropagation et faire ainsi connaissance avec les origines de l'apprentissage profond.

2.1.1 SVM

Un des algorithmes d'apprentissage automatique qui a connu le plus de succès est les SVM (Support Vector Machine). Présenté sous sa forme moderne par [Boser *et al.*, 1992] en 1992, l'algorithme consiste, dans le cas d'une classification binaire, à trouver l'hyperplan séparant les exemples de deux classes et qui maximise sa marge avec les deux classes, comme illustré dans la figure 2.2. L'algorithme est capable d'effectuer une classification linéaire, mais peut aussi effectuer une classification non-linéaire, avec une astuce appelée "kernel trick", qui consiste à transformer l'espace des caractéristiques en espace à plus grande dimension.

L'algorithme fonctionne ainsi : on considère un dataset d'exemples sous la forme $(x_1, y_1), ..., (x_n, y_n)$ où y_i vaut -1 ou 1 suivant la classe correspondant à x_i et où chaque x_i est un vecteur p-dimensionnel à valeurs réelles. Un hyperplan peut alors être défini comme l'ensemble des points vérifiant l'équation 2.1 donnée ci-dessous :

$$w^T x - b = 0 \tag{2.1}$$

où w est le vecteur normal à l'hyperplan.

Le problème d'optimisation à résoudre est alors de minimiser $||x||$ avec pour condition $y_i(w^T x - b) \geqslant 1$ pour $i = 1, ..., n$.

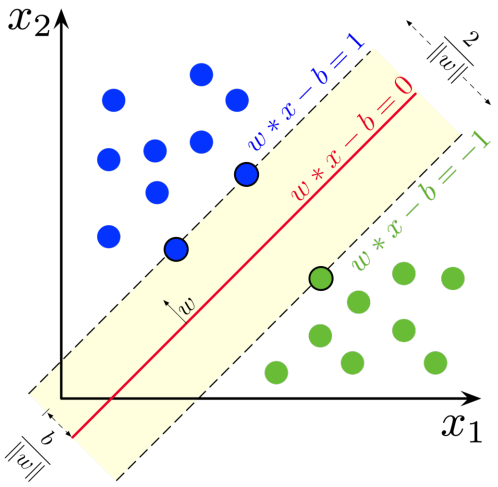


FIGURE 2.2 – Hyperplan de marge maximum et marges pour un SVM entraîné avec des exemples de deux classes. Les exemples situés aux marges sont appelés vecteurs de support [Wikipedia, 2022]

Les SVM ont été appliqués à de nombreux défis de l'IA, en particulier, en vision par ordinateur. Ils ont souvent été couplés à des extracteurs de caractéristiques pour effectuer des tâches de classification [Lin *et al.*, 2011]. Cependant la popularité des SVM a grandement diminué depuis la montée en puissance des techniques d'apprentissage profond.

2.1.2 Perceptron

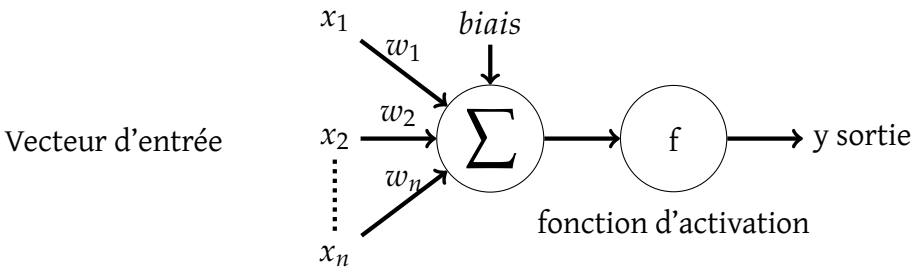


FIGURE 2.3 – Modèle du perceptron

Un perceptron est un algorithme d'apprentissage supervisé effectuant une classification binaire [Minsky et Papert, 1987]. Une somme pondérée des entrées est réalisée avant de passer le résultat à une fonction d'activation, comme illustré dans la figure 2.3. La formule du perceptron est donnée par l'équation 2.2 :

$$y = f(\sum_{i=1}^{n} w_i x_i + biais) \tag{2.2}$$

n étant le nombre d'entrées, w_i le vecteur des poids, x_i le vecteur d'entrées et y la sortie. Un perceptron seul utilisant une fonction d'activation binaire simple (par exemple, $f(x) = 1$ si $x > 0$, sinon $f(x) = 0$) est équivalent à un SVM linéaire. Un perceptron seul ne permet pas de représenter des fonctions complexes, mais si plusieurs perceptrons sont organisés en grand nombre et en plusieurs couches, leur puissance se trouve décuplée, ce qui a conduit au développement sur plusieurs décennies de l'apprentissage profond.

Dans l'équation 2.2, $f(z)$ est la fonction d'activation utilisée pour donner un comportement non-linéaire car les modèles linéaires ne sont pas assez expressifs. Plusieurs fonctions d'activations sont populaires dans les réseaux de neurones. Nous en présentons trois ci-dessous :

1. L'expression mathématique de la fonction sigmoïde est définie dans l'équation 2.3 et la figure 2.4(a) montre un graphe de la fonction sigmoïde. Elle a une symétrie centrale autour du point (0,0.5) et peut normaliser l'entrée entre 0 et 1. Récemment, la popularité de cette fonction d'activation a déclinée, en grande partie à cause du problème de disparition du gradient pour les réseaux profonds. De plus, comme c'est une fonction au centre non-zéro, cette fonction peut mener à une dérive de la covariance, les gradients oscillants pendant l'entraînement, ce qui résulte en une faible vitesse d'entraînement [Nwankpa *et al.*, 2018].

$$f(z) = sigmoïde(z) = \frac{1}{1 + e^{-z}} \tag{2.3}$$

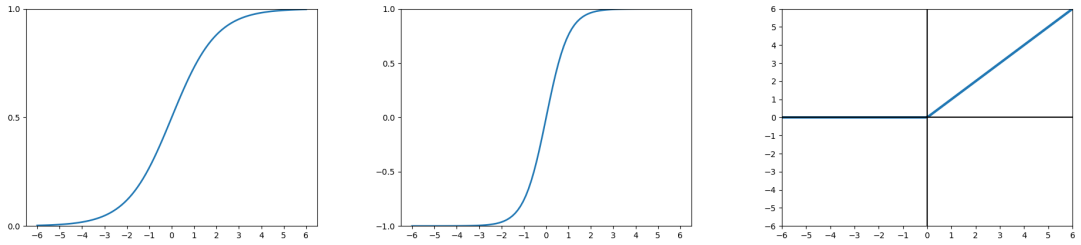
2. L'expression mathématique de la fonction tangente hyperbolique est définie dans l'équation 2.4 et la figure 2.4(b) illustre le graphe de la fonction hyperbolique. Elle est similaire à la fonction sigmoïde mais évolue en (-1,1) et est symétrique autour du point zéro, ce qui implique que le risque de dérive de la covariance est petit.

$$f(z) = tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{2.4}$$

3. La fonction Rectified Linear Unit (ReLU) met la sortie à zéro si l'entrée est négative, sinon la sortie est égale à l'entrée, comme présenté dans l'équation 2.5, et illustré dans la figure 2.4(c). Cette fonction est simple et requiert moins de calcul que les deux fonctions précédentes.

$$f(z) = relu(z) = max(0, z) \tag{2.5}$$

Comparé aux deux autres fonctions d'activations, ReLU est davantage utilisée dans les réseaux de neurones, grâce à ses avantages : tout d'abord, la sigmoïde et la tangente



(a) (b) (c)

FIGURE 2.4 – Graphes de trois fonctions d'activations (sigmoïde, tangente hyperbolique et ReLU).

hyperbolique calculent la fonction d'activation à l'aide de l'opération exponentielle, qui requiert beaucoup de temps de calcul. Quand la rétropropagation est utilisée pour calculer le gradient de l'erreur, la dérivation implique une division qui requiert un grand temps de calcul. La fonction d'activation ReLU peut donc économiser beaucoup de temps de calcul. Des expériences [Krizhevsky *et al.*, 2012] ont montré qu'un réseau utilisant ReLU est six fois plus rapide que l'équivalent utilisant la fonction tangente hyperbolique pendant l'entraînement. Ensuite, pour un réseau profond, quand la fonction sigmoïde est propagée en arrière, le gradient peut facilement disparaître. Quand la sigmoïde est proche de la région saturée, la transformation est trop lente et la dérivée tend vers 0, ce qui cause une perte d'information et gêne l'entraînement des réseaux profonds. Enfin, la fonction ReLU peut mettre la sortie de certains neurones à 0, ce qui mène à un réseau épars, réduit l'interdépendance des paramètres et diminue le problème d'over-fitting.

2.2 Apprentissage profond

L'apprentissage profond est le domaine de l'intelligence artificielle qui est devenu le plus populaire à partir des années 2010. Les techniques d'apprentissage profond constituent une sous-partie de l'apprentissage automatique, tel qu'illustré dans la figure 2.1.

2.2.1 Perceptron multi-couches

La conception et l'entraînement de réseaux de neurones artificiels multi-couches ont été formalisés en 1986 par Rumelhart et al. [Rumelhart *et al.*, 1986] à partir du perceptron décrit par Rosenblatt en 1957 [Rosenblatt, 1958]. Ces modèles permettent de développer une représentation "cachée" de leur entrée au sein de leurs couches intermédiaires, grâce à l'algorithme de rétropropagation, qui calcule les gradients de chaque paramètre du modèle multi-couche, en commençant par ceux de la dernière couche avant de remonter jusqu'à la première couche. Nous détaillerons cet algorithme plus loin dans ce chapitre. Dans cet article, les auteurs présentent l'algorithme d'entraînement qui sera

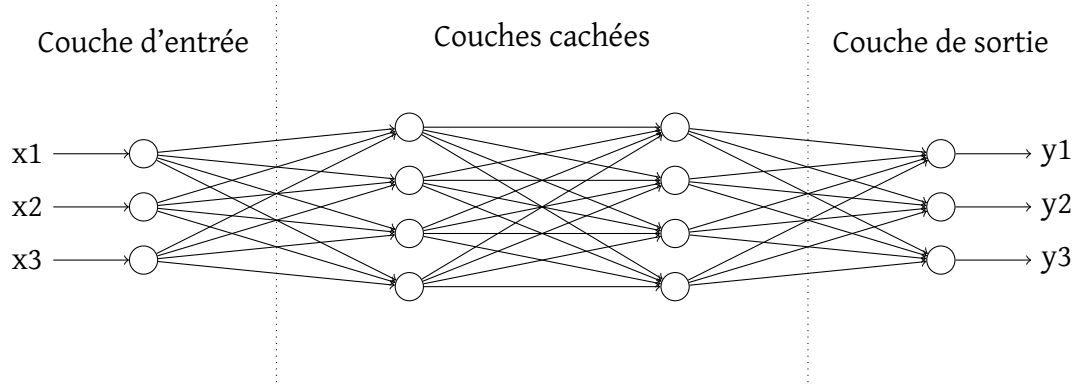


FIGURE 2.5 – Perceptron multi-couches.

nommé plus tard "descente de gradient stochastique", et qui utilise un petit nombre d'exemples de la base d'entraînement pour obtenir une approximation du gradient plutôt que l'usage du vrai gradient calculé sur toute la base de données, ce qui serait beaucoup plus lent. La figure 2.5 illustre un perceptron multi-couches.

2.2.2 Propagation avant

Dans un réseau de neurones, le processus consistant à relier les données d'une couche précédente à la couche suivante s'appelle "propagation avant", et implique de calculer la sortie de tous les neurones. Le processus de "propagation avant" peut être calculé selon l'équation 2.2 pour obtenir la valeur d'un neurone individuel avec des poids et des biais spécifique.

Pour calculer les valeurs des neurones dans chaque couche, l'équation 2.6 est utilisée :

$$X^l = f(W^{l^T} X^{l-1} + B^l) \tag{2.6}$$

Où $X^l = (x_1^l, ..., x_{n^l}^l)$ est le vecteur de n^l neurones dans la couche l. $W^l = [w_{ij}^l]$ est la matrice des poids dans la couche l, avec $1 \leqslant j \leqslant n^l, 1 \leqslant i \leqslant n^{l-1}$. $B^l = (b_1^l, ..., b_{n^l}^l)$ est le vecteur de n^l biais dans la couche l. En général, ce processus est décrit par l'équation 2.7 suivante :

$$Y = f_{w,b}(X) \tag{2.7}$$

ici, le réseau est décrit comme une fonction, où X est l'entrée, Y la sortie, et (w, b) sont les paramètres internes du réseau.

2.2.3 Rétropropagation

Pendant l'entraînement d'un réseau de neurones, les poids associés à chaque connexion entre neurones doivent être ajustés afin d'améliorer progressivement les performances

du réseau. Pour mesurer la performance du réseau, une fonction de loss est calculée en comparant la sortie du réseau à la cible qu'il est censé prédire [Feng, 2022]. Cette fonction de loss (ou fonction de perte) peut être représentée selon l'équation 2.8 :

$$C(w,b) = erreur(Y,\hat{Y}) \tag{2.8}$$

où w, b sont respectivement les vecteurs des poids et des biais du réseau (weights and biases), Y est le vecteur des valeurs réelles (ou cibles) et $\hat{Y}$ est le vecteur des valeurs prédites par le réseau.

Ensuite, l'enjeu est de calculer le gradient de la fonction de loss par rapport à chaque poids à ajuster. La méthode qui permet de calculer ce gradient s'appelle "rétropropagation". L'entraînement consiste en un processus qui va chercher à optimiser les poids et les biais pour diminuer la fonction de loss. Ceci peut être décrit par l'équation 2.9 suivante :

$$w^*, b^* = argmin_{w,b}(C(w,b)) \tag{2.9}$$

Le gradient de la fonction de loss par rapport aux poids et aux biais du réseau peut être noté ainsi :

$$\nabla C = (\nabla_w C, \nabla_b C) \tag{2.10}$$

où $\nabla_w C$ est le gradient par rapport aux poids et $\nabla_b C$ est le gradient par rapport aux biais.

Les dérivées partielles en fonction des poids et des biais sont notées comme dans les équations 2.11 et 2.12, avec X^{l-1} qui représente le vecteur des entrées de la couche l :

$$\frac{\partial C}{\partial B^l} = \delta^l \tag{2.11}$$

$$\frac{\partial C}{\partial W^l} = \delta^l X^{(l-1)} \tag{2.12}$$

Le delta de la couche de sortie est calculé comme décrit dans l'équation suivante :

$$\delta^{l_{sortie}} = \frac{\partial C}{\partial Y} f'(S^{l_{sortie}}) = \frac{\partial C}{\partial Y} f'(W^{l_{sortie}} X^{l_{sortie}-1} + B^{l_{sortie}}) \tag{2.13}$$

Les deltas des couches autres que la couche de sortie sont calculés selon l'équation 2.14 :

$$\delta^l = ((W^{l+1})^T \delta^{l+1}) \dot{f}'(S^l) \tag{2.14}$$

Ainsi, il est possible de calculer les dérivées partielles de la fonction de loss par rapport à chaque paramètre d'un perceptron multi-couche. La méthode de descente du gradient peut ensuite être appliquée pour minimiser la fonction de loss. La séquence

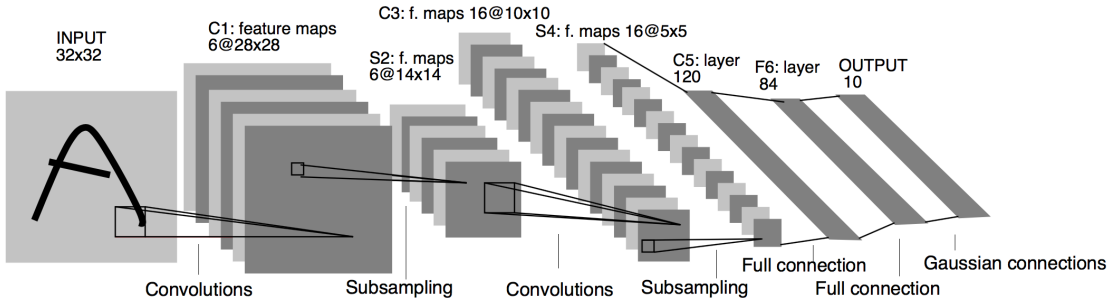


FIGURE 2.6 – LeNet : première architecture pour l'apprentissage supervisé de la reconnaissance de chiffres [LeCun *et al.*, 1989].

d'approximation $(W^l(k), B^l(k)), k \geq 0$ est calculée comme décrit dans les équations suivantes :

$$B^l(k+1) = B^l(k) + \eta\delta^l \tag{2.15}$$

$$W^l(k+1) = W^l(k) + \eta\delta^l X^{l-1}(k) \tag{2.16}$$

où k représente le nombre d'itérations, $\eta > 0$ est le taux d'apprentissage et $X^{l-1}(k)$ est l'entrée de la couche l.

Aujourd'hui, la backpropagation n'est qu'un cas particulier de l'application de la règle de la chaîne à n'importe quelle fonction composée différentiable. Ceci est implémenté dans chaque bibliothèque d'apprentissage profond en tant que mode inverse de la dérivation automatique.

2.2.4 Réseaux convolutifs

L'architecture des réseaux convolutifs (ou CNN) est inspirée par la biologie du cortex visuel humain. Dès 1968, des travaux montrent que les chats et les singes disposent de neurones qui s'activent pour des régions plus ou moins petites du champ visuel, qu'ils appellent champs réceptifs [Hubel et Wiesel, 1968]. Plusieurs décennies plus tard, en vision par ordinateur, l'utilisation de neurones disposant d'un champ réceptif plus ou moins grand et organisés en hiérarchie est proposée pour l'apprentissage non supervisé de la reconnaissance des nombres [Fukushima, 2004]. En 1989, Lecun et al [LeCun *et al.*, 1989] appliquent pour la première fois, à la tâche de reconnaissance de chiffres, un réseau de neurones d'architecture convolutive, entraîné par rétropropagation avec un algorithme de descente de gradient stochastique. La figure 2.6 illustre cette architecture pionnière.

L'expression générale de l'opération de convolution est définie selon l'équation 2.17 :

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{2.17}$$

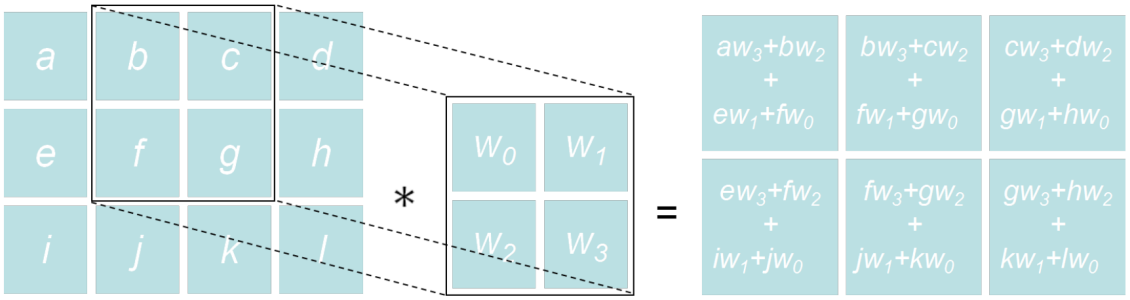


FIGURE 2.7 – L'opération de convolution 3x3, notée $*$ [Feng, 2022].

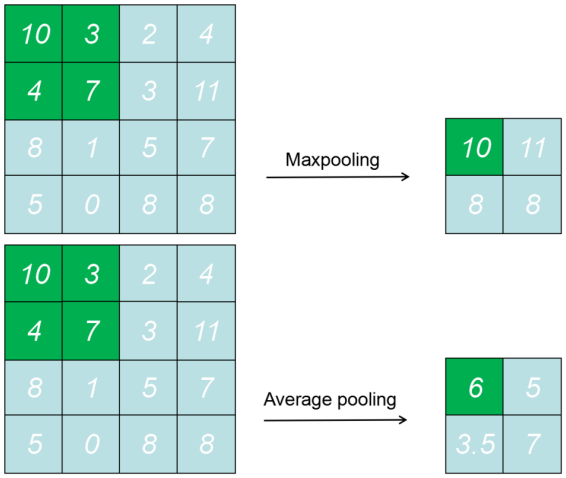


FIGURE 2.8 – Diagramme des fonctions "Max pooling" et "Average pooling".

où I est l'entrée 2D, K est le kernel 2D, et S(i,j) est la sortie filtrée [Feng, 2022]. L'opération de convolution calcule la moyenne pondérée du champ réceptif, comme illustré dans la figure 2.7.

Les réseaux convolutifs partagent généralement les poids entre les neurones d'un même plan de caractéristiques, ce qui donne aux réseaux convolutifs, la propriété d'équivariance par translation. C'est à dire que si l'image en entrée est translatée, les caractéristiques en sortie du réseau seront translatées proportionnellement de la même manière. Cela rend également les réseaux convolutifs épars, ce qui améliore grandement leur efficacité.

L'autre opération qui a mené aux premiers succès des réseaux convolutifs est le pooling. La fonction de pooling remplace la sortie du réseau à un endroit particulier par une statistique agrégée des sorties voisines de cet endroit. Par exemple, la fonction Max pooling [Zhou et Chellappa, 1988] donne la valeur maximum dans la zone rectangulaire autour de l'endroit considéré. D'autres fonctions de pooling populaires existent comme la moyenne ou la moyenne pondérée en fonction de la distance au pixel central considéré. La figure 2.8 illustre les fonctions "Max pooling" et "Average pooling".

Peu importe la fonction de pooling utilisée, le pooling est invariant par translation,

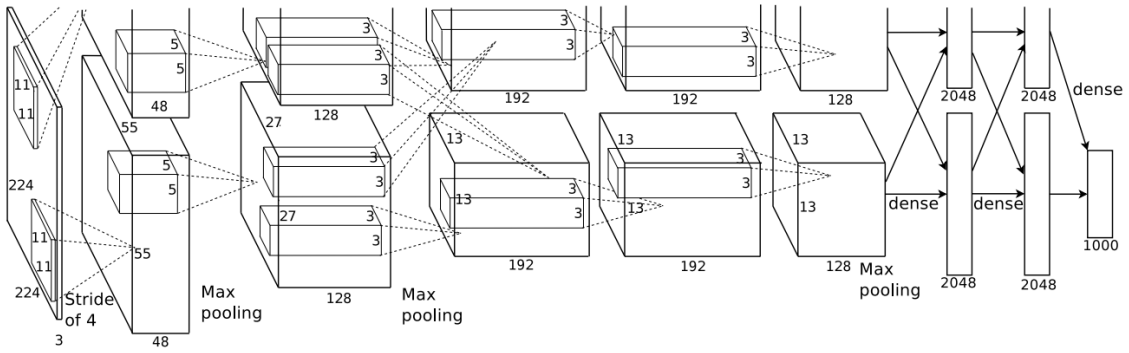


FIGURE 2.9 – Architecture d'AlexNet [Krizhevsky *et al.*, 2012].

c'est à dire qu'avec un petit déplacement dans l'entrée, la plupart des sorties après la fonction de pooling ne changent pas. Comme le pooling synthétise la valeur de chaque voisin, cela permet au nombre d'unités d'être bien inférieur à ceux utilisant la convolution ou les unités entièrement connectées. Cette réduction dans le nombre d'unités améliore l'efficacité et réduit le besoin de stocker des paramètres. Le pooling est également important pour beaucoup de tâches pour permettre la gestion de différentes tailles d'entrée. Par exemple, lorsque l'on veut classifier des images de différentes tailles, l'entrée de la couche de classification doit être de taille fixe. Ceci est généralement obtenu en ajustant la taille de la région concernée par le pooling pour que la couche de classification reçoive toujours le même nombre de caractéristiques statistiques peu importe la taille de l'entrée.

En 2011, Ciresan et al. [Ciresan *et al.*, 2011] ont présenté une implémentation de CNN capable de bénéficier de la puissance de calcul en parallèle des GPU (Graphical Processing Units), obtenant des résultats qui atteignent l'état de l'art de l'époque en reconnaissance de chiffres sur MNIST [Lecun *et al.*, 1998], et en reconnaissance d'objets sur les datasets NORB [LeCun *et al.*, 2004] et CIFAR10 [Krizhevsky, 2009].

Les CNNs ont ensuite fait leur preuve sur la compétition de classification d'objets ImageNet ILSRVC en 2012, avec AlexNet [Krizhevsky *et al.*, 2012], un réseau convolutif accéléré sur GPU et régularisé avec la technique "dropout". L'architecture est illustrée dans la figure 2.9. C'est à partir de ce moment que la plupart des défis en vision par ordinateur ont eu recours de manière intensive aux réseaux convolutifs. Cela a conduit à de très grands progrès sur la plupart des benchmarks. Les réseaux entraînés pour la classification d'objets sur le dataset ImageNet servent aujourd'hui de référence et sont souvent affinés sur d'autres tâches en aval.

Les progrès sur ImageNet se sont poursuivis, avec en particulier l'utilisation de l'architecture Resnet [He *et al.*, 2016]. Les auteurs proposent d'utiliser des connections sautées entre des blocs qu'ils appellent "résiduels", illustrés dans la figure 2.10 et définis par

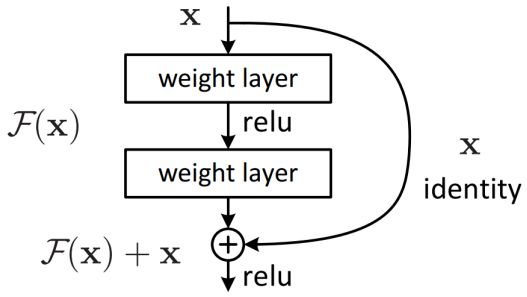


FIGURE 2.10 – Bloc de base pour un réseau résiduel. "relu" est un raccourci pour "Rectifier Linear Unit".

l'équation suivante :

$$y = F(x, W_i) + x \tag{2.18}$$

où $F(x, W_i)$ représente la fonction résiduelle à apprendre. L'architecture est constituée d'un empilement de ces blocs résiduels, illustré dans la figure 2.11. Ceci revient à une simplification par rapport aux architectures précédentes, tout en permettant de créer des réseaux plus profonds, et d'atteindre de nouveaux niveaux de performance. Cette architecture permet également de palier au problème de dégradation du gradient qui affecte les réseaux de neurones profonds classiques. Dans la suite de ce manuscrit, nous utilisons comme backbone ResNet-50 (une architecture résiduelle avec 50 couches) et ResNet-101 (une architecture résiduelle plus profonde, avec 101 couches) [He *et al.*, 2016].

2.2.5 Optimisation

Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD), est un type d'algorithme d'optimisation par descente du gradient. Dans les problèmes d'optimisation où l'on cherche à minimiser une fonction objectif convexe ou possédant de multiples minimums locaux similaires, la descente de gradient (méthode générale pour la résolution des systèmes d'équations simultanées) est couramment utilisée. C'est le cas pour la grande majorité des architectures de réseaux de neurones profonds modernes. Cependant, il est beaucoup trop coûteux de calculer le gradient entier sur de grandes bases de données, c'est pourquoi la méthode SGD est favorisée. En effet, elle se contente d'un échantillon de la base de données pour calculer une estimation du gradient à chaque pas d'entraînement. Cette méthode repose sur l'espoir que l'effet de bruit induit par l'échantillonnage s'estompe au fil des pas d'entraînements et ne gêne pas la convergence vers l'optimum local. Empiriquement, cet espoir est vérifié dans la quasi-totalité des cas, ce qui permet à la descente de gradient stochastique d'être, du fait de sa simplicité, l'algorithme d'optimisation de réseau de neurone le plus populaire. De plus, l'usage du SGD est quasiment systématiquement

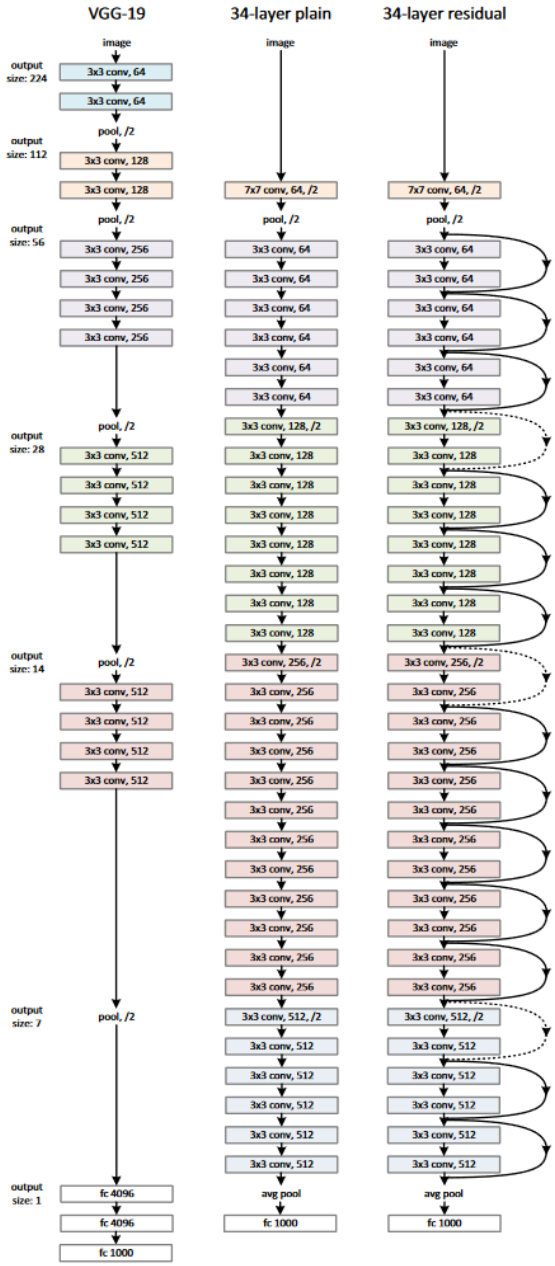


FIGURE 2.11 – Exemples d'architectures pour ImageNet. Gauche : Le modèle VGG-19 [Simonyan et Zisserman, 2015] comme référence. Milieu : un réseau classique avec 34 couches. Droite : un réseau résiduel avec 34 couches.

couplé avec un momentum, qui permet au déplacement dans la surface de loss de gagner en inertie et d'atteindre plus rapidement le minimum local, tout en ignorant des irrégularités dans la surface. Dans son usage standard, un learning rate constant est utilisé pour tous les paramètres à optimiser.

Adam

L'algorithme Adam [Kingma et Ba, 2015], originellement présenté en 2015, est une alternative à SGD pour l'optimisation de fonctions objectifs. Il a été conçu pour combiner les avantages des algorithmes AdaGrad [Duchi *et al.*, 2011], qui fonctionne bien avec des gradients épars, et RMSProp [Hinton, 2012], qui fonctionne bien dans des cas où la fonction objectif n'est pas constante. Adam utilise et met à jour le learning rate indépendemment pour chaque paramètre du modèle, à chaque pas de temps, grâce au calcul des moments de premier et de second ordre. L'algorithme est aujourd'hui de plus en plus utilisé à la place de SGD, car il est empiriquement plus robuste et efficace, et ne nécessite pas de grands efforts pour choisir ses paramètres.

2.2.6 Fonctions de loss

Dans de nombreux articles, le choix de la fonction de loss a montré son intérêt. C'est pourquoi, plusieurs fonctions de loss provenant de [Bruckert *et al.*, 2021] seront considérées dans ce manuscrit.

MSE loss La fonction de loss MSE (Mean Squared Error) mesure l'erreur moyenne au carré de la prédiction par rapport à la cible. Elle est définie par la formule suivante :

$$MSE(Y, \hat{Y}) = \frac{1}{N} \sum_{i=0}^{N} (Y - \hat{Y})^2 \tag{2.19}$$

où Y représente la valeur cible et $\hat{Y}$ est la valeur prédite.

CC loss

La fonction de loss CC (Correlation Coefficient) mesure la corrélation linéaire entre la prédiction et la cible. Elle est définie par la formule suivante :

$$CC(Y, \hat{Y}) = \frac{\sigma(Y, \hat{Y})}{\sigma(Y)\sigma(\hat{Y})} \tag{2.20}$$

où $\sigma(Y)$ est la variance de Y et $\sigma(Y, \hat{Y})$ est la covariance de Y et $\hat{Y}$.

Focal Loss

La Focal Loss a été proposée pour gérer les cas où la classe de background est bien plus présente que la classe de foreground [Lin *et al.*, 2017b]. Il s'agit d'une modification de la fonction de loss binary cross entropy. Elle est définie selon l'équation 2.21 et est configurable au travers du paramètre γ.

$$FL(Y, \hat{Y}) = -\sum_{i=1}^{N}((1-\hat{Y}_i^{\gamma}) \times Y_i log(\hat{Y}_i) + \hat{Y}_i^{\gamma}(1-Y_i) log(1-\hat{Y}_i)) \tag{2.21}$$

2.3 Bases de données

En apprentissage automatique, les bases de données sont indispensables pour deux usages : l'entraînement, et l'évaluation. Pendant l'entraînement, les algorithmes d'apprentissage tirent des échantillons de la base de données afin d'améliorer progressivement leurs performances. L'évaluation, par une mesure de performance, permet de vérifier l'efficacité d'un algorithme ou d'une méthode d'entraînement, et de comparer différents algorithmes.

2.3.1 Bases de données d'images

ImageNet

Le dataset ImageNet [Deng *et al.*, 2009], publié en 2009, a été à la source d'une grande partie des avancées récentes en traitement d'images. Il est constitué de millions d'images, collectées avec des requêtes sur des moteurs de recherche d'images, et annotées à l'aide d'Amazon Mechanical Turk [Sorokin et Forsyth, 2008]. Il existe différentes catégories organisées selon une structure hiérarchique provenant de WordNet [Miller, 1995], illustrée dans la figure 2.12. On y retrouve 12 "subtrees" (mammifère, oiseau, poisson, reptile, amphibien, véhicule, fourniture, instrument de musique, formation géologique, outil, fleur, fruit) subdivisés en 5247 catégories, chacune contenant en moyenne 600 images. Le subtree le plus représenté est celui des mammifères, qui contient 1170 catégories et 862K images.

MS-COCO

MS-COCO est une grande base de données d'images, annotées pour la segmentation d'instance (la tâche d'assigner pour chaque pixel d'une image, une classe d'objet ou bien la classe du background) à l'aide d'Amazon Mechanical Turk. Les instances d'objets sont classées parmi 91 catégories, regroupées en 11 super-catégories (personnes et

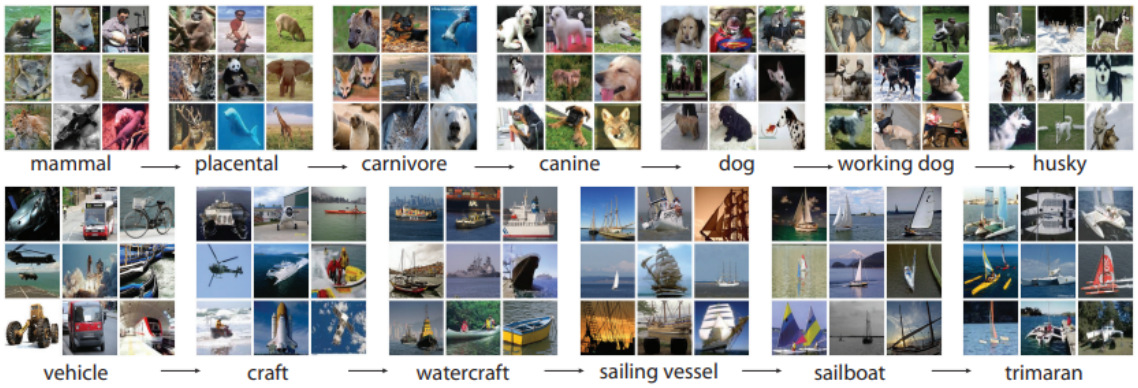


FIGURE 2.12 – Exemples de deux branches de catégories d'Imagenet, de leur racine jusqu'à leur feuille [Deng *et al.*, 2009].

FIGURE 2.13 – Douze images extraites du dataset MS-COCO [Lin *et al.*, 2014].

accessoires, animaux, véhicules, objets d'extérieur, sports, ustensiles de cuisine, nourriture, fourniture, appareils, électronique, objets d'intérieur). La catégorie "personnes" est la plus représentée en termes de nombre d'instances. En 2017, le dataset a publié des annotations de points clés de squelettes de personnes, couvrant 17 points clés (nez, oeil gauche, oeil droit, oreille gauche, oreille droite, épaule gauche, épaule droite, coude gauche, coude droit, poignet gauche, poignet droit, hanche gauche, hanche droite, genou gauche, genou droit, cheville gauche, cheville droite). Les exemples sont séparés en trois ensembles "train", "validation, et "test". Un échantillon de quelques images présentes dans MS-COCO est illustré dans la figure 2.13.

2.3.2 Bases de données vidéos

Posetrack

FIGURE 2.14 – Douze images extraites du dataset PoseTrack [Andriluka *et al.*, 2018].

PoseTrack18 [Andriluka *et al.*, 2018], illustré dans la figure 2.14, contient des séquences vidéos de quelques dizaines de frames chacunes, extraites du dataset populaire MPII Human Pose Dataset [Andriluka *et al.*, 2014]. Le dataset est là aussi subdivisé en trois ensembles "train", "validation" et "test". L'ensemble "train" contient 550 vidéos, l'ensemble "validation" en contient 550, et l'ensemble "test" en compte 375. Les vidéos sont annotées avec des poses de personnes, caractérisées par 15 parties du corps (tête, nez, cou, épaules, coudes, poignets, hanches, genoux, chevilles). Chaque pose annotée est associée à un identifiant de suivi propre à une personne dans la séquence vidéo. Les vidéos présentent de grandes variations en termes de poses, de mouvements, de contextes, et des caractéristiques qui rendent le traitement difficile, comme par exemple, des personnes partiellement ou complètement occultées, des changements d'échelles, ou des foules denses. La majorité des séquences contiennent entre 41 et 151 frames, ce qui correspond à quelques secondes en temps réel.

2.4 Conclusion

Dans ce chapitre, nous avons tout d'abord détaillé le domaine de l'apprentissage automatique en le situant dans le cadre plus global de l'intelligence artificielle et en présentant deux de ses principaux algorithmes, SVM et le perceptron. Ensuite, nous avons expliqué les notions principales de l'apprentissage profonds, en présentant ses bases avec le perceptron multi-couches, capable d'apprendre des représentations internes grâce à

l'algorithme de rétropropagation, puis en expliquant les réseaux de neurones convolutifs (CNN) et leur histoire. De plus, nous avons présenté les deux algorithmes d'optimisation les plus populaires, SGD et Adam, et les fonctions de loss qui seront utilisées dans ce manuscrit. Enfin, nous avons présenté les principales bases de données qui seront utilisées dans ce manuscrit.

Chapitre 3

Segmentation d'instance sur image Fisheye

Dans ce chapitre, nous présentons une méthode d'augmentation de données pour adapter Mask R-CNN à la segmentation d'instance en image fisheye, tout en conservant de bonnes performances en image rectilinéaire. Nous expliquons d'abord notre motivation, puis nous présentons un état de l'art qui donne le contexte dans lequel nos travaux se situent, ensuite nous expliquons la méthode que nous proposons, puis nous présentons les expériences réalisées pour évaluer notre méthode, et enfin, nous concluons.

3.1 Motivation

Dans les environnements de transports, et plus particulièrement dans le ferroviaire avec l'arrivée du futur train autonome, la détection et le suivi de personnes est une brique cruciale pour des objectifs de sureté et de sécurité. Parmi les usages potentiels, nous pouvons citer la détection d'intrusion, de mouvement de foule, d'agitation, de violence, d'agression ou d'évanouissement.

Il existe différents types d'objectifs de caméra qui sont utilisés pour la surveillance vidéo. En général, les objectifs grand angle sont favorisés car ils permettent de surveiller un grand espace avec une seule caméra. Les objectifs fisheye avec un angle de vue de 180° sont très utiles pour la surveillance vidéo. La figure 3.1 illustre une image acquise avec un objectif fisheye. Cependant, les caméras à objectif fisheye obtiennent cet angle de vue en sacrifiant la perspective rectilinéaire. Les lignes droites dans la scène qui ne passent pas par le centre de l'image subissent une distorsion qui les transforme en lignes courbes. Cette distorsion est radiale (c.-à-d. que les points sont déplacés vers le centre de l'image) et est dite "en barillet" (voir figure 3.2). De plus, la distorsion subie par un objet dépend de son emplacement dans la scène. Un algorithme de surveillance utilisant une caméra à objectif fisheye doit donc être capable de gérer ces particularités pour profiter de son grand champ de vision.

Cependant, à ce jour, il n'existe pas de base de données d'images fisheye de taille suffisante pour entraîner des modèles de réseaux de neurones convolutifs (CNN) profonds. La création d'un tel dataset est un projet très long et coûteux. Or, les CNN entraînés sur des bases de données d'images classiques (c.-à-d. rectilinéaires) apprennent à reconnaître des motifs présents uniquement dans ce type d'images. Ces motifs seront déformés dans les images qui présentent des distorsions en barillet. En conséquence, de tels CNN obtiendront des performances fortement amoindries sur les images fisheye.

Les solutions à ce problème se rangent en deux catégories : 1) correction des distorsions présentes dans l'image. 2) adaptation de l'algorithme pour traiter directement les images fisheye.

La correction des distorsions, aussi appelée un-warping ou dewarping, consiste généralement à utiliser un modèle simplifié de la caméra afin d'inverser l'effet. Cette opération est coûteuse en temps, et cause une perte d'information aux bords de l'image, nécessitant une interpolation à cause de la faible densité de pixels à ces endroits. Ce problème est illustré dans la figure 3.3 où le modèle de caméra à objectif fisheye équidistant [Kannala et Brandt, 2006] est utilisé. Dès lors qu'il faille trouver un équilibre entre la qualité de la correction, le champ de vision, les artefacts, et le temps de calcul, ces défauts réduisent l'intérêt d'utiliser des caméras à objectif fisheye.

Adapter un algorithme de détection et de suivi de personnes pour traiter directement les images fisheye est difficile à cause du manque de données d'entraînement. Une solution est de créer des images fisheye synthétiques à partir d'images rectilinéaires. C'est l'approche que nous avons choisi.

Cependant, dans le futur train autonome, le serveur de calcul qui traitera les informations provenant des caméras sera embarqué à l'intérieur du train et disposera donc d'une puissance de calcul et d'une mémoire fixe et limitée. De plus, deux types de caméras seront utilisées, une caméra grand angle à objectif fisheye et une autre dite standard. Dans ce contexte, utiliser des algorithmes différents en fonction du type de caméra impose un besoin additionnel de mémoire et rend le système plus complexe. Il est donc intéressant de disposer d'un algorithme de détection capable de traiter les deux types d'images, sans modification spécifique. Pour accomplir cet objectif, nous proposons une technique d'augmentation de données qui transforme pendant l'entraînement une partie des images rectilinéaires en images à effet fisheye (FE).

3.2 État de l'art

Les progrès récents dans la tâche de segmentation d'instance ont été permis grâce à la publication de datasets de grande taille et de bonne qualité, adaptés pour l'entraînement de réseaux de neurones convolutifs. Les principaux datasets utilisés sont ImageNet

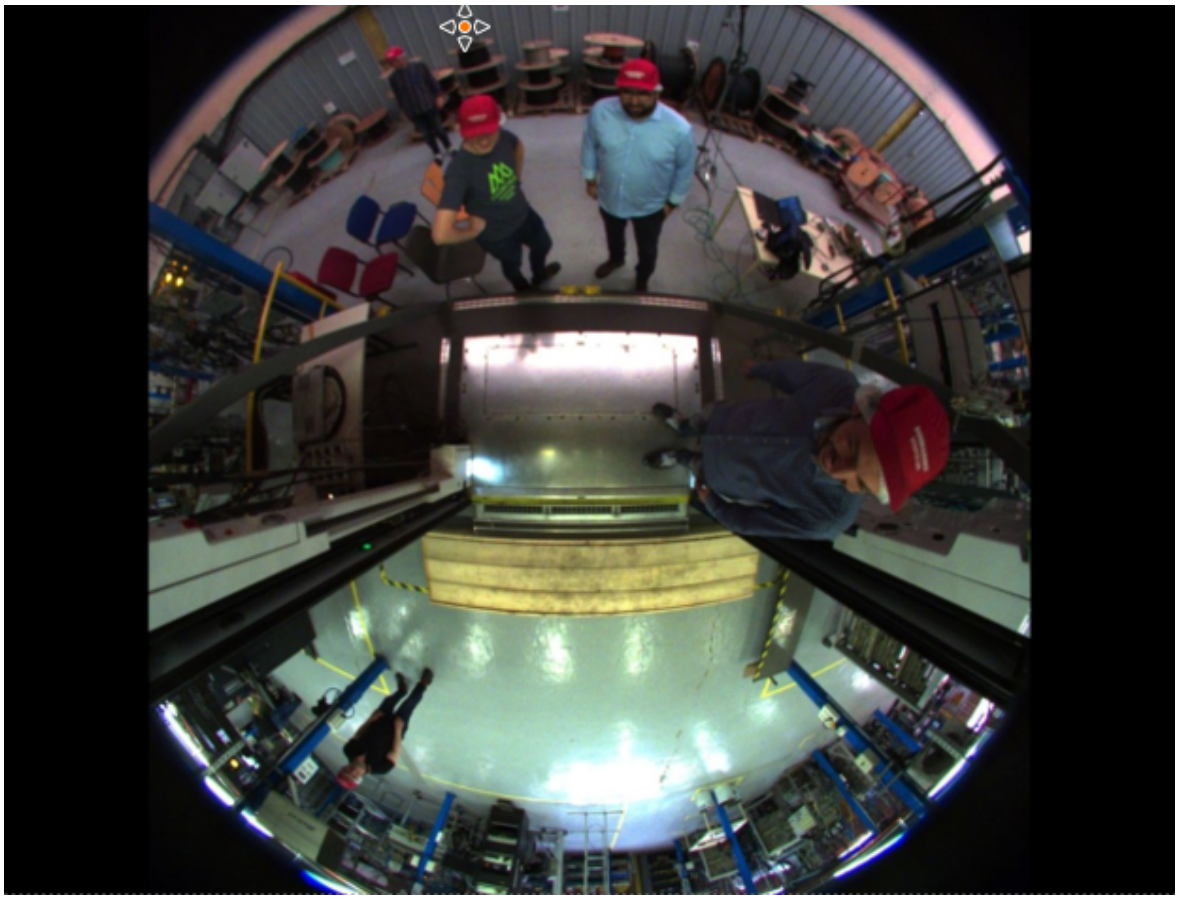
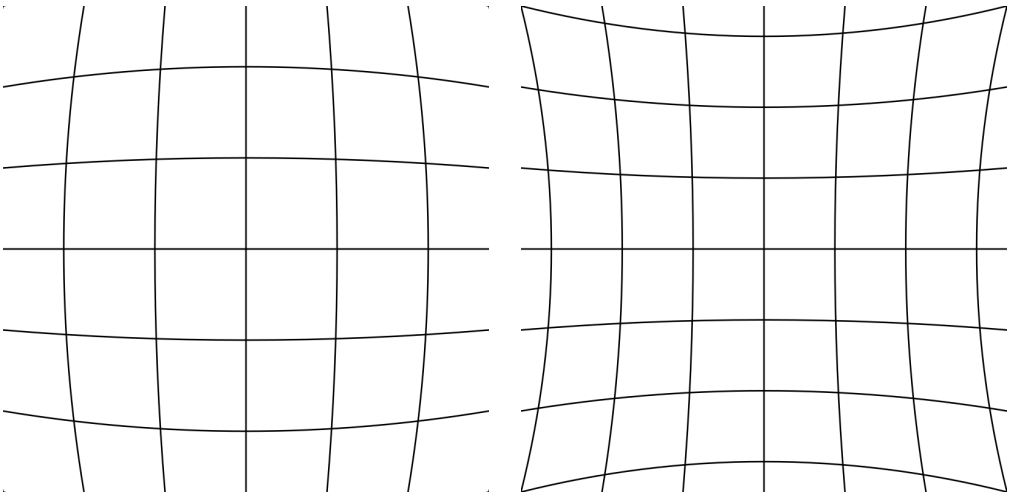


FIGURE 3.1 – Image Fisheye présentant des distorsions en barillet à cause du grand angle de vue de l'objectif.



(a) Distorsion en barillet (b) Distorsion en coussinet

FIGURE 3.2 – Distorsions radiales.

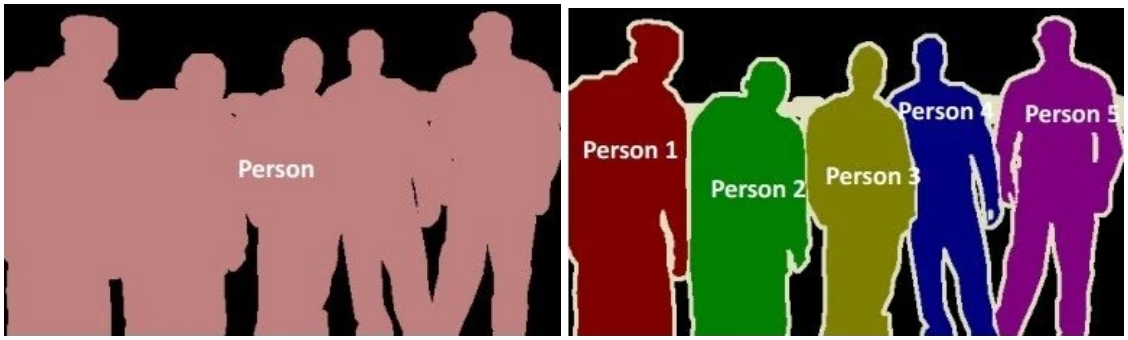
(a) Image fisheye (b) Image corrigée

FIGURE 3.3 – Correction fisheye.

[Deng *et al.*, 2009], MS-COCO [Lin *et al.*, 2014] et CityScape [Cordts *et al.*, 2016]. Cependant, ces datasets ne contiennent pas d'images fisheye. Le manque de dataset fisheye a été comblé récemment par WoodScape [Yogamani *et al.*, 2019], OmniScape [Sekkat *et al.*, 2020] et "GO Stanford" [Hirose *et al.*, 2018]. Cependant, WoodScape ne contient que 10000 images fisheye annotées pour la segmentation d'instance, acquises à partir d'une caméra placée sur le toit d'un véhicule, il est donc plus petit que MS-COCO et moins diversifié. OmniScape est également peu diversifié, car c'est un dataset construit à partir d'images capturées dans le jeu vidéo GTA V et le simulateur open source CARLA, conçus pour simuler des environnements urbains en extérieur. Enfin "Go Stanford" contient uniquement des images d'intérieur de bâtiments, et manque donc également de diversité pour permettre l'entraînement efficace d'un réseau convolutif.

Les images fisheye sont utilisées pour de multiples tâches de vision par ordinateur. Hirose et al. [Hirose *et al.*, 2018] utilisent par exemple un GAN pour prédire la traversabilité dans des images fisheye afin de guider un robot. Zhang et Al. [Zhang *et al.*, 2016] évaluent la pertinence des caméras grand angle pour la tâche d'odométrie visuelle. Caruso et al. [Caruso *et al.*, 2015] proposent une méthode de SLAM (Simultaneous Localization And Mapping) adaptée pour l'utilisation d'une caméra fisheye.

En traitement d'image, la segmentation sémantique est une tâche qui consiste à donner, pour chaque pixel de l'image, la classe d'objets à laquelle il appartient. La segmentation d'instance demande en plus, à ce que chaque pixel qui n'appartient pas au fond soit assigné à une instance d'objet. La distinction entre ces deux catégories de segmentation est illustrée en figure 3.4. Plusieurs algorithmes de segmentation sémantiques adaptés aux images fisheye ont ainsi été développés. En 2019, Saez et al. [Saez *et al.*, 2019] ont proposé une architecture de CNN modifiée, et ont élaboré une stratégie d'augmentation de données FE (Fisheye Effect), dans le contexte des voitures autonomes. Blott et al. [Blott *et al.*, 2018] est l'article le plus en lien avec notre problématique. Les auteurs proposent une méthode d'augmentation de données FE qui repose sur un modèle de pro-



(a) Segmentation sémantique (b) Segmantation d'instance

FIGURE 3.4 – Différence entre la segmentation sémantique et la segmentation d'instance [Odemakinde,].

jection sphérique, et qui permet d'entraîner un CNN de segmentation sémantique sur des images FE créées à partir d'images rectilinéaires. Plus spécifiquement, 25 transformations fisheye différentes sont créées grâce au modèle de projection fisheye introduit par [Christopher, 2009] qui diffèrent selon 6 DoF (Degrees of Freedom). Cela permet l'entraînement d'un VGG16 sur des exemples tirés de MS-COCO réduit à 16 classes (au lieu de 80), chaque image étant augmentée grâce aux 25 transformations. Le modèle résultant de l'entraînement avec augmentation FE est comparé, sur 50 vraies images fisheye, à une référence entraînée sans augmentation, et à une baseline entraînée avec des augmentations rectilinéaires (rotation, translation, renversement vertical ou horizontal), ce qui permet de conclure positivement sur l'utilité de l'augmentation FE.

Cependant, la segmentation d'instance à partir d'images fisheye est un sujet de recherche moins exploré. Saito et al. [Mamoru Saito *et al.*, 2010] ont utilisé, en 2010, une technique de soustraction d'arrière-plan, en acceptant comme contraintes une caméra statique et des personnes droites. Ces suppositions ne sont malheureusement pas valides dans le cadre de notre application qui est la détection et le suivi de personnes à l'intérieur d'un train. En 2019, Wang et al. [Wang *et al.*, 2019] ont entraîné un Mask R-CNN en ajoutant une classe "Fisheye-Person" et une augmentation de données par rotation aléatoire comprise entre 0 et 360 degrés.

L'algorithme le plus souvent utilisé comme référence dans la tâche de segmentation d'instance est Mask R-CNN [He *et al.*, 2017]. Il est constitué d'une architecture de type Faster R-CNN [Ren *et al.*, 2015], auquel Mask R-CNN rajoute une branche qui segmente chaque objet détecté au sein d'une boîte englobante. Ce type d'architecture a été présenté pour la première fois par Girshick et al. [Girshick *et al.*, 2014]. Ils furent les premiers à appliquer les nouvelles méthodes basées sur les CNN, à la tâche de détection d'objet, qui était jusque là dominée par des méthodes utilisant les descripteurs SIFT [Lowe, 2004] et HOG [Dalal et Triggs, 2005]. R-CNN génère environ 2000 propositions de régions en uti-

lisant l'algorithme selective search [Uijlings *et al.*, 2013], puis calcule pour chaque région proposée, un vecteur de caractéristiques, avant de classifier chaque région à l'aide d'un SVM spécifique pour chaque catégorie. Le temps nécessaire pour calculer les vecteurs de caractéristiques rend cette méthode lente, ce qui motiva les auteurs à présenter ensuite Fast R-CNN [Girshick, 2015]. Cet algorithme effectue une seule inférence d'un CNN plutôt qu'une par région candidate, grâce à une couche "RoI pooling" qui permet d'obtenir une carte de caractéristique spécifique à un candidat à partir des caractéristiques calculées par un réseau pour toute l'image. Faster R-CNN a ensuite été proposé, et remplace la selective search par un RPN (Region Proposal Network) qui permet de partager les mêmes caractéristiques pour effectuer les propositions d'objets et les classifier. Faster R-CNN applique tout d'abord un backbone qui permet d'extraire des caractéristiques en utilisant "RoI pooling" pour chaque fenêtre glissante. Ces caractéristiques sont ensuite utilisées par une tête "reg" (regression de boîte englobante) et "cls" (classification). C'est à partir de l'architecture Faster R-CNN que Mask R-CNN a été proposé. L'architecture de ce dernier est donnée dans la figure 3.6. Mask R-CNN rajoute une tête de segmentation en plus des têtes "reg" et "cls". De plus, les auteurs proposent une nouvelle méthode pour agréger les caractéristiques appellée "RoIAlign", illustrée dans la figure 3.5. Cela permet un alignement correct des caractéristiques de chaque pixel de la boîte englobante et donc une plus grande précision, ce qui s'avère crucial pour obtenir de bonnes performances. Mask R-CNN prédit un masque de segmentation pour chaque classe d'objet, ce qui permet d'éviter la compétition entre les classes et de découpler la prédiction de classe de la prédiction du masque. Une partie cruciale de ce type d'algorithme est le backbone qui extrait des caractéristiques de l'image. Mask R-CNN utilise comme extracteur de caractéristiques un réseau résiduel ResNet-50, Resnet-101 ou ResNeXt-101, et couple ce backbone avec un FPN (Feature Pyramid Network) [Lin *et al.*, 2017a] qui utilise des connexions latérales pour obtenir des caractéristiques multi-échelles. Le backbone est pré-entraîné pour la tâche de classification sur ImageNet, avant d'être entraîné end-to-end sur la base de données de segmentation d'instance MS-COCO. La loss utilisée est multi-tâche, suivant la formule décrite ci-dessous :

$$L = L_{cls} + L_{box} + L_{mask} \tag{3.1}$$

avec L_{cls} la loss de classification, L_{box} la loss de boîte englobante, et L_{mask} la loss du masque de segmentation. L_{cls} et L_{box} sont identiques à celles définies dans [Girshick, 2015], tandis que L_{mask} est la loss de cross-entropy binaire moyenne.

La méthode que nous proposons dans la suite de ce chapitre est la première à s'intéresser à la segmentation d'instance d'images fisheye, la plupart des travaux précédents s'intéressant à la segmentation sémantique. Nous utilisons une méthode d'augmentation de données similaire à celle utilisée par Blott et al. [Blott *et al.*, 2018] pour la segmentation sémantique. De plus, nous sommes les premiers à s'intéresser à obtenir un

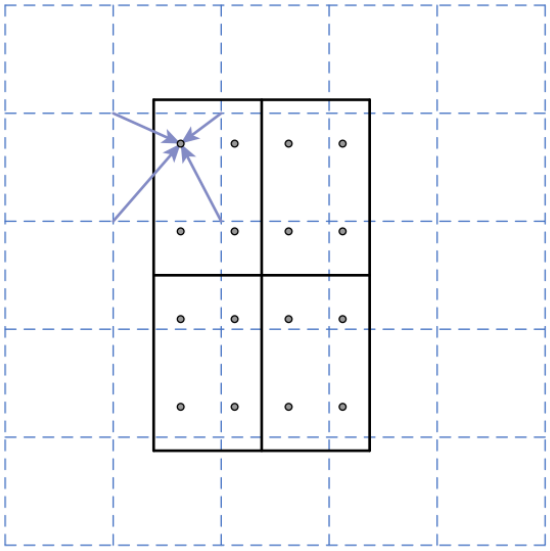


FIGURE 3.5 – RoIAlign : une interpolation bilinéaire est utilisée pour calculer la valeur de caractéristique pour 4 lieux d'échantillonage.

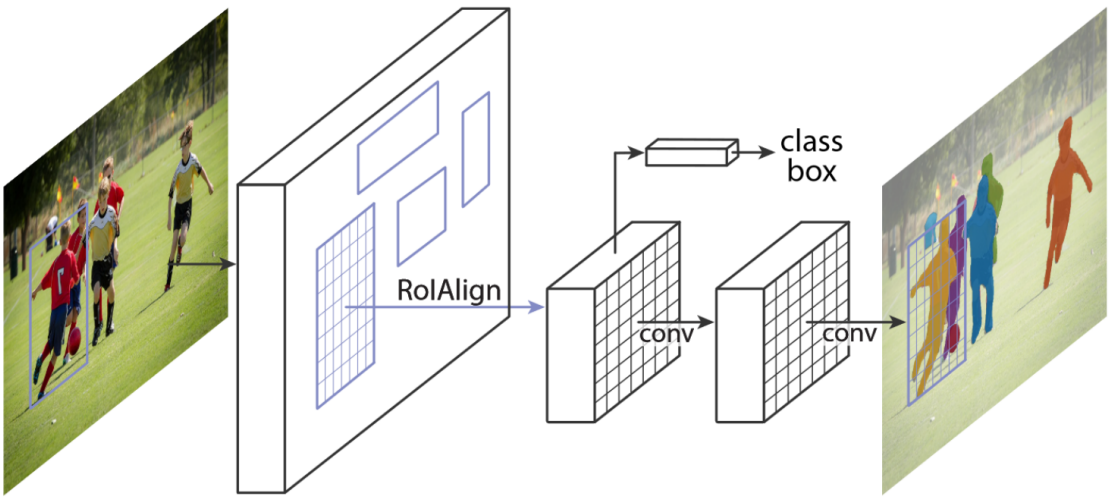


FIGURE 3.6 – Architecture simplifiée de Mask R-CNN.

compromis efficace en terme de performances à la fois sur des images rectilinéaires et avec la même méthode sur des images fisheye ou grand angle.

3.3 Méthode

L'approche que nous proposons est basée sur le fait qu'une image rectilinéaire prise grâce à une projection centrale et l'annotation correspondante peuvent être transformées en image FE (Fisheye Effect) en exploitant un modèle de projection commun. Nous utilisons comme référence une architecture de segmentation d'instance existante, Mask R-CNN [He *et al.*, 2017]. Nous cherchons à modifier la manière dont elle est entraînée afin de la rendre capable de traiter des images fisheye, tout en conservant de bonnes performances sur des images rectilinéaires, ce qui constitue un défi qui n'a pas encore été exploré dans l'état de l'art. L'approche d'adaptation au domaine fisheye consiste à ré-entraîner Mask R-CNN sur sa base de données (COCO2017), avec une augmentation de données FE dont le principe est illustré dans la figure 3.7. Cette augmentation utilise le même modèle de projection que [Blott *et al.*, 2018], originellement décrit dans [Christopher, 2009], et dont nous rappelons brièvement le fonctionnement ci-après :

1. Le plan image est translaté, on applique une rotation autour de l'axe z, et une mise à l'échelle.

2. Les points du plan image sont projetés sur la sphère unitaire.

$$(\overrightarrow{\mathcal{X}_s})_{\mathcal{F}_m} = \frac{\overrightarrow{\mathcal{X}}}{\|\overrightarrow{\mathcal{X}}\|} = (X_s, Y_s, Z_s) \tag{3.2}$$

3. Le référentiel $\mathcal{F}_m$ est ensuite déplacé d'une distance selon l'axe z pour donner le référentiel $\mathcal{F}_p$.

$$(\overrightarrow{\mathcal{X}_s})_{\mathcal{F}_p} = (X_s, Y_s, Z_s + \xi) \tag{3.3}$$

4. Les points sont projetés sur le plan image normalisé π_{m_u}.

$$m_u = (\frac{X_s}{Z_s + \xi}, \frac{Y_s}{Z_s + \xi}, 1) \tag{3.4}$$

5. Des distorsions radiales sont effectuées selon le modèle introduit par Brown [Sanz-Ablanedo *et al.*, 2010].

$$\overrightarrow{m_d} = \overrightarrow{m_u} + D(\overrightarrow{m_u}, V) \tag{3.5}$$

où D est la fonction de distorsion avec coefficients de distorsion V.

6. Projection finale avec la matrice de projection K $\begin{pmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{pmatrix}$ où f est la focale et u_0, v_0 sont les coordonnées du point principal. Les coordonnées finales sur le plan image fisheye (π_p) sont finalement :

$$\overrightarrow{p} = K \cdot \overrightarrow{m_d} \tag{3.6}$$

Ce modèle de projection est utilisé pour créer des transformations pré-calculées afin de ne pas augmenter la durée de l'entraînement. Cette transformation diffère catégoriquement des augmentations rectilinéaires (par exemple la mise à l'échelle et la translation) car les objets sont déformés différemment en fonction de leur distance au point central. Le principe est illustré dans la figure 3.8. L'entraînement est effectué sur des échantillons rectilinéaires ou avec augmentation FE selon un ratio d'augmentation R. L'évaluation est quant à elle réalisée sur trois types de datasets : rectilinéaire, fisheye réelle ou bien fisheye synthétique (obtenues par l'application de l'augmentation FE sur des images rectilinéaires). Nous avons créé deux ensembles de transformations FE. Le premier contient 35 transformations complexes illustrées dans la figure 3.9(a), conçues pour être grandement différenciées dans ces paramètres de transformation sphérique :

- Translations dans l'intervalle [0;0.5] en échelle normalisée ;
- Rotations quelconques autour de l'axe z ;
- Mise à l'échelle dans l'intervalle [50%;120%] ;
- Distorsions radiales dans l'intervalle [-0.5,-1.0] ;
- Paramètre de projection ξ dans l'intervalle [0;1].

Le deuxième ensemble de transformations illustré dans la figure 3.9(b) contient 8 transformations FE plus simples, différenciées uniquement par leur rotation, selon un pas de $\frac{\pi}{4}$. Les images d'illustrations sont obtenues par application de l'effet fisheye artificiel sur une seule image multicolore source.

Le ratio d'augmentation R est défini afin d'appliquer l'augmentation FE à une portion des exemples d'entraînement. Nous montrons plus loin dans ce chapitre, la détermination de la valeur optimale de ce ratio. Chaque image utilisée dans l'entraînement a une probabilité égale à R d'être sélectionnée pour l'augmentation. Quand une image est choisie pour l'augmentation FE, une transformation spécifique est choisie selon une loi uniforme parmi les transformations pré-calculées afin d'optimiser le temps de calcul.

L'entraînement de Mask R-CNN est effectué avec les paramètres par défaut du framework Detectron [Girshick *et al.*, 2018], et quelques ajustements comme une plus grande

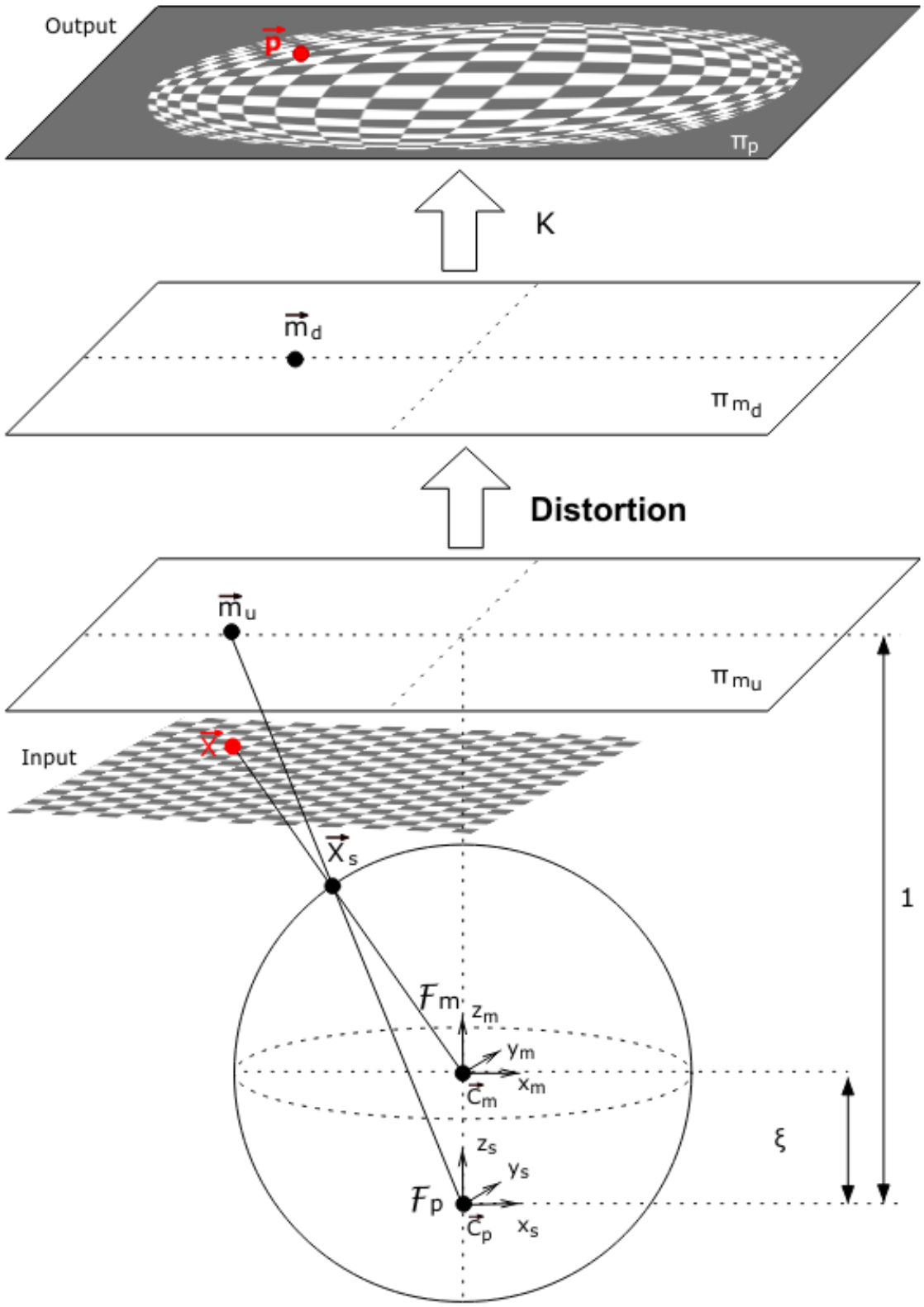


FIGURE 3.7 – Modèle de projection décrit dans [Christopher, 2009] et utilisé pour notre problématique.

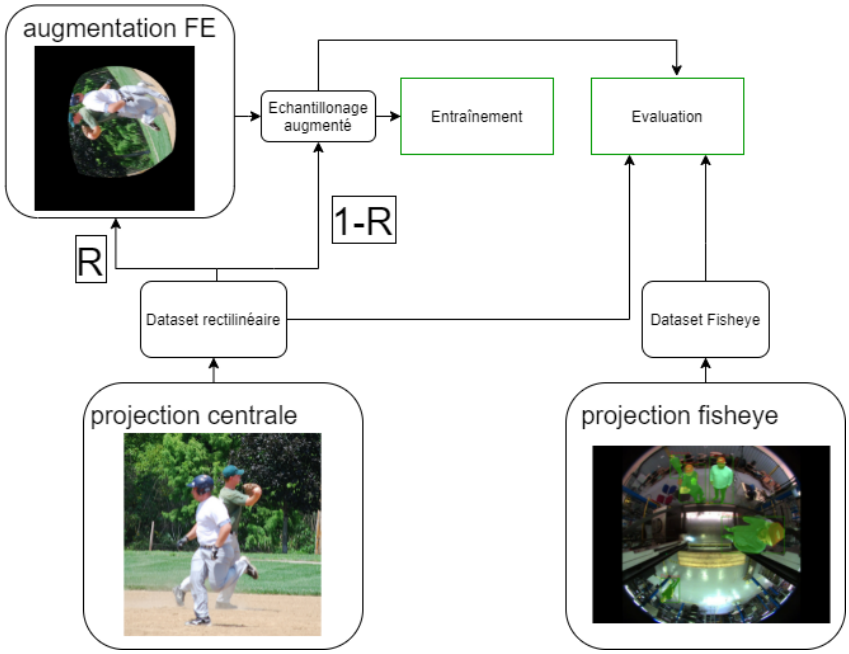
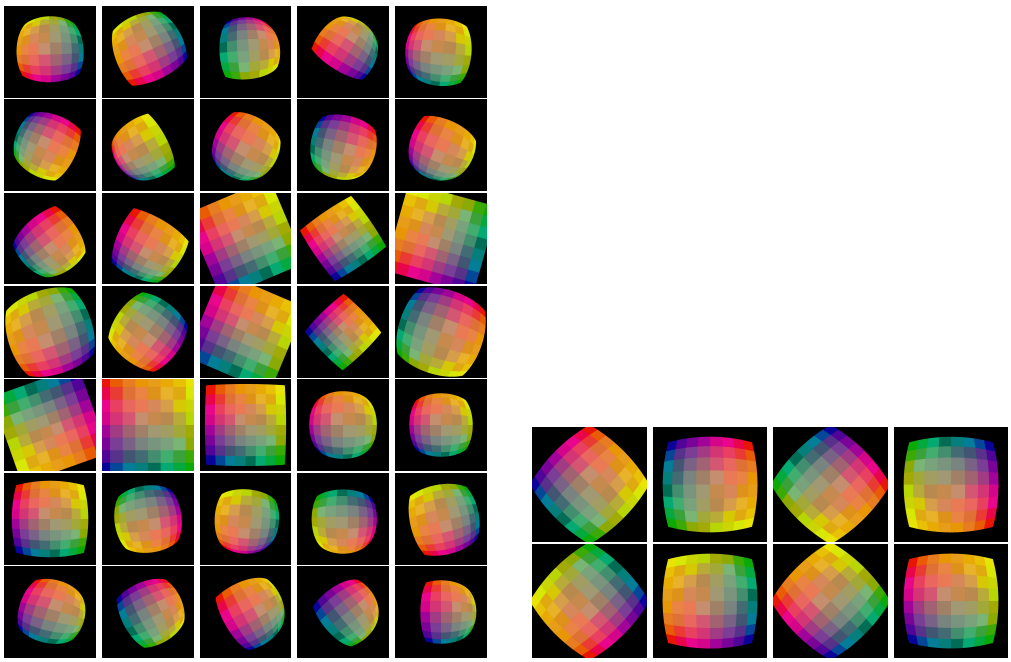


FIGURE 3.8 – Stratégie d'augmentation FE. Le nombre d'images d'entraînement est augmenté avec l'utilisation du ratio R. Les performances sont évaluées sur les datasets fisheye et rectilinéaires.



(a) Ensemble de 35 transformations complexes (b) Ensemble de 8 transformations simples

FIGURE 3.9 – Les deux ensembles de transformations utilisés dans notre approche.

taille de batch par GPU, et moins de pas d'entraînement, afin d'obtenir de bonnes performances sur notre matériel (GPU Nvidia Tesla V100). Deux programmes d'entraînement différents sont utilisés : un programme long pour l'architecture ResNet-101, et un plus

court pour l'architecture ResNet-50. Le programme long prévoit 90k pas d'entraînement et un learning rate de 0.02 qui est divisé par 10 aux pas 50k et 70k. Le programme court prévoit 40k pas d'entraînement et un learning rate de 0.02 qui est divisé par 10 aux pas 20k et 30k.

3.4 Résultats

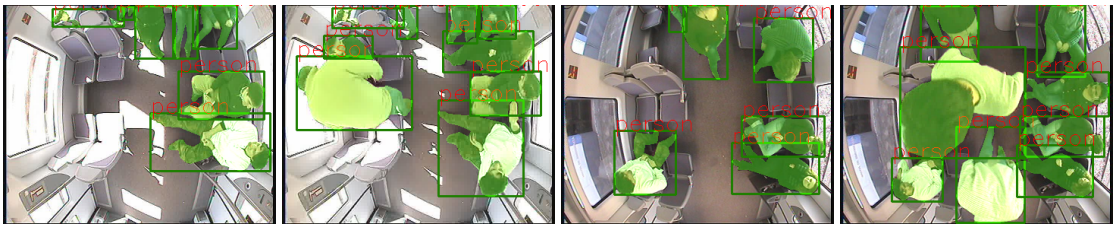
3.4.1 Datasets d'évaluation

Les résultats de détection sont évalués sur le subset de validation de COCO2017 (ie. COCO2017val) ainsi que sur trois datasets fisheye dédiés spécifiquement à notre application (i.e. valBOSS, trainDoor et trainDoorAug), et dans lesquels nous avons segmenté les personnes à l'aide du logiciel CVAT [Manovich et Sekachev, 2019]. Le subset valBOSS contient 60 frames tirées de deux séquences du dataset BOSS [Velastin et Gómez-Lira, 2017] (Les images ont été acquises dans un train en mouvement avec une caméra grand angle. Elles présentent des distorsions en barillet et les contraintes de l'embarqué avec la présence d'ombres portées et d'un background défilant). Le dataset trainDoor a été acquis par l'IRT Railenium dans le cadre du projet train autonome. Un environnement de portes du futur train autonome a été reproduit dans un entrepôt du constructeur Alstom Crespin. Le dataset contient 121 frames acquises avec une caméra à objectif fisheye et simule le passage d'usagers aux portes du train. Pour constituer le dataset trainDoorAug, nous avons utilisé un renversement vertical sur trainDoor pour doubler le nombre d'exemples à savoir 242 frames. Des exemples de ces datasets sont présentés dans la figure 3.10. Pour les trois datasets valBOSS, trainDoor et trainDoorAug, notons que seul le label "person" est considéré.

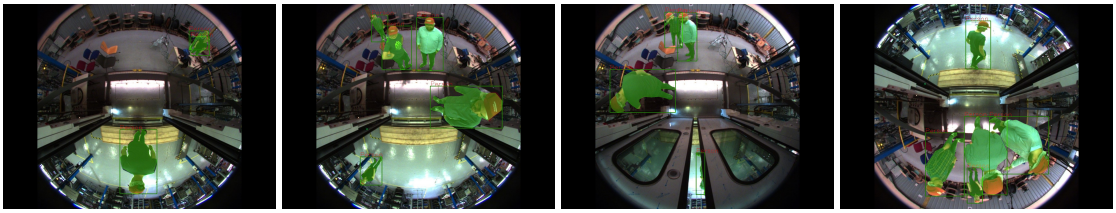
Nous évaluons la performance de notre stratégie avec les outils officiels de MS COCO. Nous considérons des métriques tirées de [Lin *et al.*, 2014] basées sur la précision (i.e. le rapport entre le nombre de vrais positifs (TP) et le nombre de vrais et faux positifs (TP+FP)). Ces métriques sont les suivantes :

- APall : la précision moyenne obtenue pour différents seuils de confiance allant de 0.50 à 0.95 par pas de 0.05 ;
- AP50 : la précision pour un seuil de 50% ;
- APL : la précision pour des objets de grande taille ;
- APM : la précision pour des objets de taille moyenne ;
- APS : la précision pour des objets de petite taille.

(a) Images annotées du dataset COCO2017



(b) Images annotées du dataset valBOSS



(c) Images annotées du dataset trainDoor

FIGURE 3.10 – Images annotées des datasets COCO2017, valBOSS et trainDoor.

3.4.2 Pré-entraînement sur image rectilinéaire

Nous avons entraîné deux modèles en utilisant l'augmentation FE. Dans le premier modèle, seul les poids du backbone sont initialisés par des poids pré-entraînés sur ImageNet [Deng *et al.*, 2009]. Les poids dans les branches de classification, de prédiction de boîtes englobantes et de segmentation sont initialisés aléatoirement. Dans le deuxième modèle, tous les poids sont initialisés à partir de ce que nous nommerons par la suite la référence1 à savoir Mask R-CNN pré-entraînée sur MS COCO. Dans le tableau 3.1, nous montrons qu'après un entraînement et une évaluation sur COCO2017val augmenté avec 35 transformations FE, le deuxième modèle gagne 4% en APall, 6% en AP50, 4% en AP75, 2% en APS, 4% en APM et 7% en APL par rapport au premier. Nous en concluons que les apprentissages de Mask R-CNN en domaine rectilinéaire sont utiles en transfer learning dans le domaine FE, c'est à dire qu'ils produisent de meilleurs résultats qu'une initialisation aléatoire. Dans la suite du manuscrit, tous les modèles seront donc initialisés selon la deuxième méthode, c.-à-d. pré-entraînés sur le dataset rectilinéaire MS COCO.

	APall	AP50	AP75	APS	APM	APL
ImageNet	12	22	12	5	15	20
COCO2017	**16**	**28**	**16**	**7**	**19**	**27**

Tableau 3.1 – Impact d'une initialisation avec backbone ImageNet vs modèle complètement entraîné sur COCO2017, évalué sur COCO2017val avec augmentation FE.

3.4.3 Ratio d'augmentation FE

En utilisant l'augmentation FE pour la totalité des images, le modèle n'est plus exposé aux images rectilinéaires, ce qui diminue ses performances sur ce type d'images. Afin d'utiliser l'augmentation FE tout en conservant de bonnes performances sur des images rectilinéaires, nous avons expérimenté différents ratios d'augmentation et un backbone ResNet50 sur COCO2017val. Les résultats sont données dans la figure 3.11. Lorsque nous augmentons le ratio d'augmentation FE nous pouvons observer une augmentation des performances allant de 6% à 15.9% sur les images fisheye et une diminution des performances allant de 37.7% à 29% sur les images rectilinéaires. Nous concluons alors qu'un ratio de 50% est un bon compromis car il apporte une amélioration de 9% en domaine FE, et ne diminue les performances rectilinéaires que de 2%. Ce choix de ratio a été confirmé en l'utilisant pour l'entraînement d'un modèle Mask R-CNN avec backbone ResNet-101 et en l'évaluant sur COCO2017val.

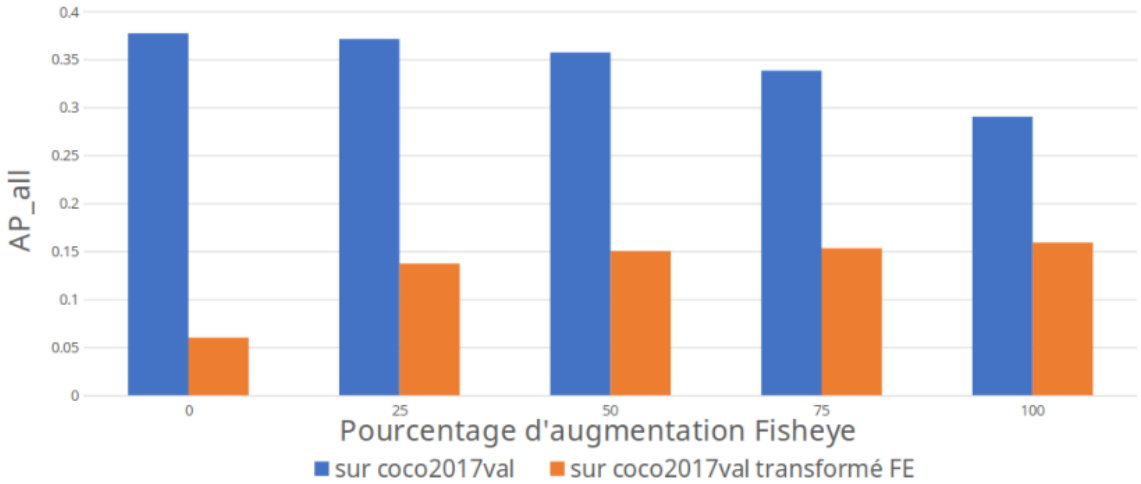


FIGURE 3.11 – Evolution de APall sur COCO2017val rectilinéaire (Bleu) et transformé FE (Orange) en fonction du ratio d'augmentation FE pendant l'entraînement.

Le tableau 3.2 montre les performances sur COCO2017val d'un Mask R-CNN avec un backbone ResNet-101 et 50% d'augmentation FE. Ce modèle dont l'entraînement a été augmenté avec un ratio de 50% a obtenu 39.5% en APall contre 40.1% pour la référence1.

	APall	**AP50**	**AP75**	**APS**	**APM**	**APL**
Référence1 (Mask R-CNN pré-entraîné sur MS COCO)	**40.1**	**61.9**	**44.0**	**22.6**	**43.6**	**52.6**
50% FE augmentation	39.5	60.7	43.2	22.1	43.5	51.7

Tableau 3.2 – Performance d'un Mask R-CNN ResNet-101 avec augmentation FE de 50% vs Référence1 sur COCO2017val.

Nous constatons une tendance similaire pour les autres métriques. Cette différence est jugée négligeable, et confirme que le modèle conserve de bonnes performances en domaine rectilinéaire.

3.4.4 Evaluation sur datasets dédiées : valBOSS, trainDoor et trainDoorAug

Après avoir évalué les performances de l'approche proposée sur les images rectilinéaires de COCO2017, nous avons démarré une série de tests visant à confirmer les bonnes performances des paramètres choisis sur nos datasets dédiés au contexte ferroviaire embarqué : valBOSS, trainDoor et trainDoorAug. Le tableau 3.3 illustre les résultats obtenus.

	APall	**AP50**	**AP75**	**APM**	**APL**
Dataset valBOSS					
Référence1 (Mask R-CNN pré-entraîné sur MS COCO)	18.0	43.9	12.1	**26.6**	10.0
50% FE augmentation	18.0	**46.0**	**12.8**	25.1	**15.1**
Dataset trainDoor					
Référence1 (Mask R-CNN pré-entraîné sur MS COCO)	55.8	78.5	66.3	49.7	59.9
50% FE augmentation	**70.3**	**90.6**	**83.6**	**59.1**	**77.6**
Dataset trainDoorAug					
Référence1 (Mask R-CNN pré-entraîné sur MS COCO)	43.5	64.7	50.4	31.5	52.2
50% FE augmentation	**67.0**	**90.7**	**79.0**	**54.7**	**74.5**

Tableau 3.3 – Performance d'un Mask R-CNN ResNet-101 entraîné avec 50% d'augmentation FE vs Référence1 sur les trois datasets FE dédiés au contexte ferroviaire embarqué.

Nous remarquons une légère amélioration des performances avec la méthode proposée sur le dataset valBOSS, de l'ordre de 2.1% en AP50, 0.9% en AP75 et 5.1% en APL, mais une diminution de 1.5% en APM. Ces tendances ne sont pas suffisamment significatives pour pouvoir tirer des conclusions. En revanche, une réelle amélioration est constatée sur trainDoor et encore davantage sur trainDoorAug. En effet, sur le dataset trainDoor, on remarque un avantage de 14.5% en APall, 12.1% en AP50, 17.3% en AP75,

9.4% en APM, et 17.7% en APL. Sur le dataset trainDoorAug, la différence est encore plus marquante avec une amélioration de 23.5% en APall, 26.0% en AP50, 28.6% en AP75, 23.2% en APM, et 22.3% en APL. Notons que ce dernier dataset est aussi plus difficile car l'augmentation de renversement vertical fait que la moitié des personnes présentes sont "à l'envers" (i.e. tête en bas, pieds en haut) et l'autre moitié "à l'endroit" (i.e. tête en haut, pieds en bas). L'orientation a une importance car les CNN ne sont pas invariants par rotation [Marcos *et al.*, 2016]. C'est vraisemblablement la raison pour laquelle la différence de performance entre la référence1 entraînée sur des personnes droites dans COCO2017 et le modèle entraîné avec augmentation FE (impliquant différentes rotations) est plus élevée sur trainDoorAug (voir figure 3.12). Cette évaluation est donc biaisée en faveur de l'augmentation FE, qui contient des rotations.

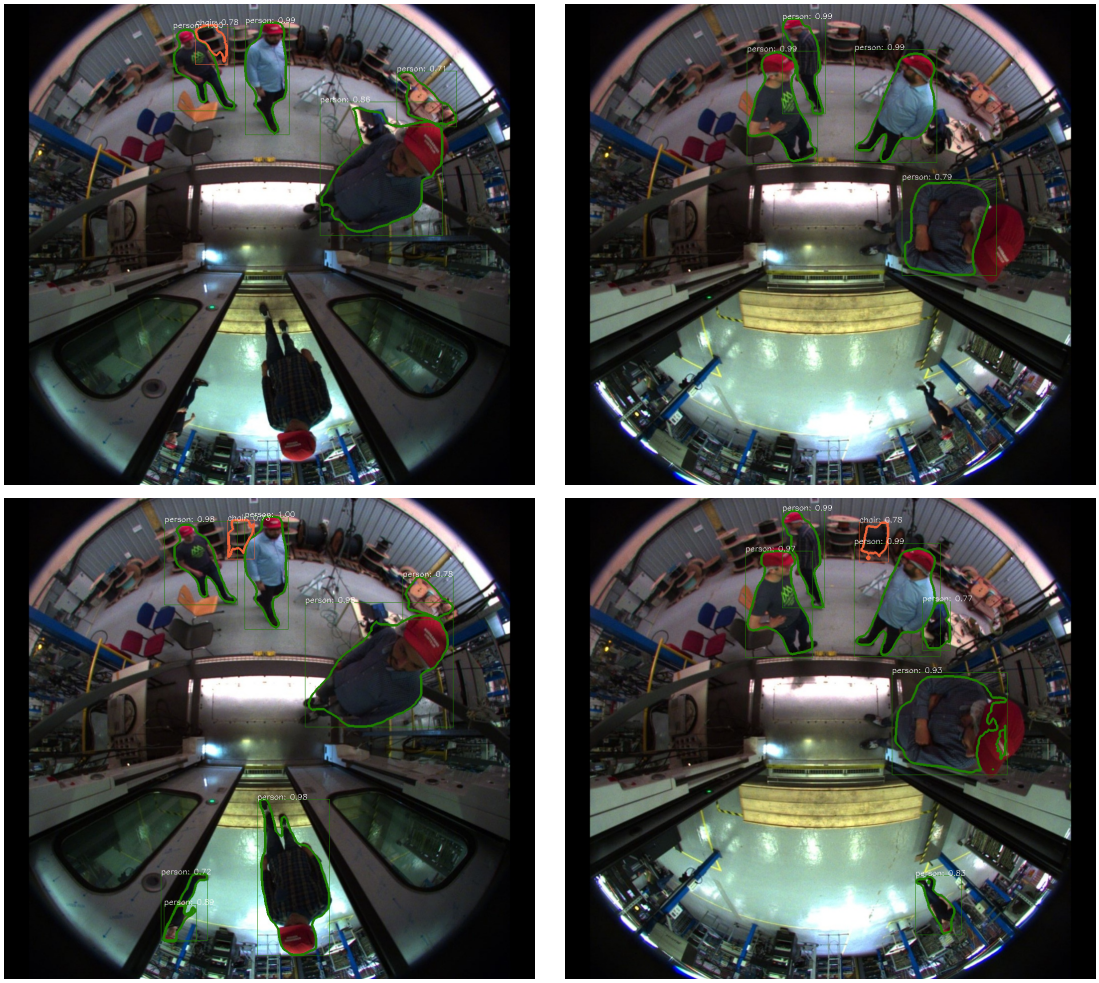


FIGURE 3.12 – Première ligne : résultats d'inférence avec Mask R-CNN ResNet-101. Deuxième ligne : résultats d'inférence avec Mask R-CNN ResNet-101 avec augmentation 35FE.

3.4.5 Augmentation FE vs renversement vertical

Afin de disposer d'une référence plus "pertinente", nous avons ensuite entraîné un modèle Mask R-CNN avec un backbone ResNet-50 (au lieu de ResNet-101, pour des questions de temps de traitement) et une augmentation de données avec renversement vertical selon un ratio de 50%. Le tableau 3.4 illustre les résultats obtenus sous le nom "halfvflip". Les résultats montrent qu'une augmentation de renversement vertical est suffisante pour améliorer les performances sur des images fisheye sur le dataset trainDoorAug. En effet, on observe une augmentation de 17% APall, 18.9% AP50, 18.9% AP75, 22.4% APM, 14.3% APL par rapport à la référence1 (Mask R-CNN pré-entraîné sur MS COCO) sur le dataset trainDoorAug. De plus, l'utilisation d'une augmentation 35FE permet d'améliorer les performances par rapport au renversement vertical avec un gain en APall de 1.9% sur valBOSS et de 1.4% sur trainDoorAug.

	APall	**AP50**	**AP75**	**APM**	**APL**
Dataset valBOSS					
Référence1 (Mask R-CNN pré-entraîné sur MS COCO)	14.6	38.1	9.3	21.5	8.8
Référence2 (Mask R-CNN entraîné avec halfvflip)	12.0	35.2	6.6	19.1	8.5
35FE transformations	13.9	39.7	7.8	20.6	9.2
8FE transformations	**22.8**	**52.8**	**17.8**	**29.6**	**9.5**
Dataset trainDoorAug					
Référence1 (Mask R-CNN pré-entraîné sur MS COCO)	43.6	67.5	50.5	31.5	51.0
Référence2 (Mask R-CNN entraîné avec halfvflip)	60.6	86.4	69.4	**53.9**	65.3
35FE transformations	61.0	87.6	71.7	51.2	67.0
8FE transformations	**62.0**	**87.7**	**74.2**	51.4	**68.5**

Tableau 3.4 – Performance d'un Mask R-CNN ResNet-50 entraîné avec 50% d'augmentation FE (avec deux ensembles de transformations différents) comparé à deux références et sur deux datasets FE.

3.4.6 Choix des ensembles de transformations

Étant donné les résultats précédents, nous nous sommes questionnés sur la nécessité d'utiliser un ensemble de 35 transformations complexes, sachant qu'une augmentation simple permet d'apporter une amélioration substantielle. En particulier, considérant les 35 transformations FE, la rotation semble être le paramètre le plus important. Nous avons donc entraîné un modèle avec une augmentation FE utilisant un nouvel ensemble de transformations simples contenant seulement 8 rotations différentes avec un effet fisheye minimal. Comme précédemment, nous évaluons ce modèle sur nos deux bases de données dédiées. Les résultats sont illustrés dans le tableau 3.4. Nous pouvons observer

une nette amélioration de l'ordre de 8.2% en APall, 14.7% en AP50, 8.5% en AP75, 8.1% en APM, et 0.7% en APL sur le dataset BOSS par rapport à la référence1 (un Mask R-CNN pré-entraîné sur MS COCO). L'amélioration est plus légère sur le dataset trainDoorAug en comparaison avec la référence2 (Mask R-CNN entraîné avec augmentation de renversement vertical pour 50% des exemples d'entraînement), mais qui est déjà supérieure de 17% en APall par rapport à la référence1 sur laquelle était basée notre comparaison précédente. Nous obtenons ainsi un gain de 1.4% en APall, de 1.3% en AP50, de 4.8% en AP75 et de 3.2% en APL malgré une diminution de 2.5% en APM, et un avantage de 3.2% en APL. Ces résultats montrent qu'une grande partie du défi associé aux images fisheye est dû aux différentes rotations d'objets qu'elles contiennent, et qu'il n'est pas nécessaire d'utiliser différents paramètres de transformation FE dans les augmentations. L'ensemble de 8 transformations FE semble donc être plus adapté pour améliorer les performances sur les images fisheye réelles. Ces résultats sont illustrés qualitativement dans les figures 3.13 à 3.16.

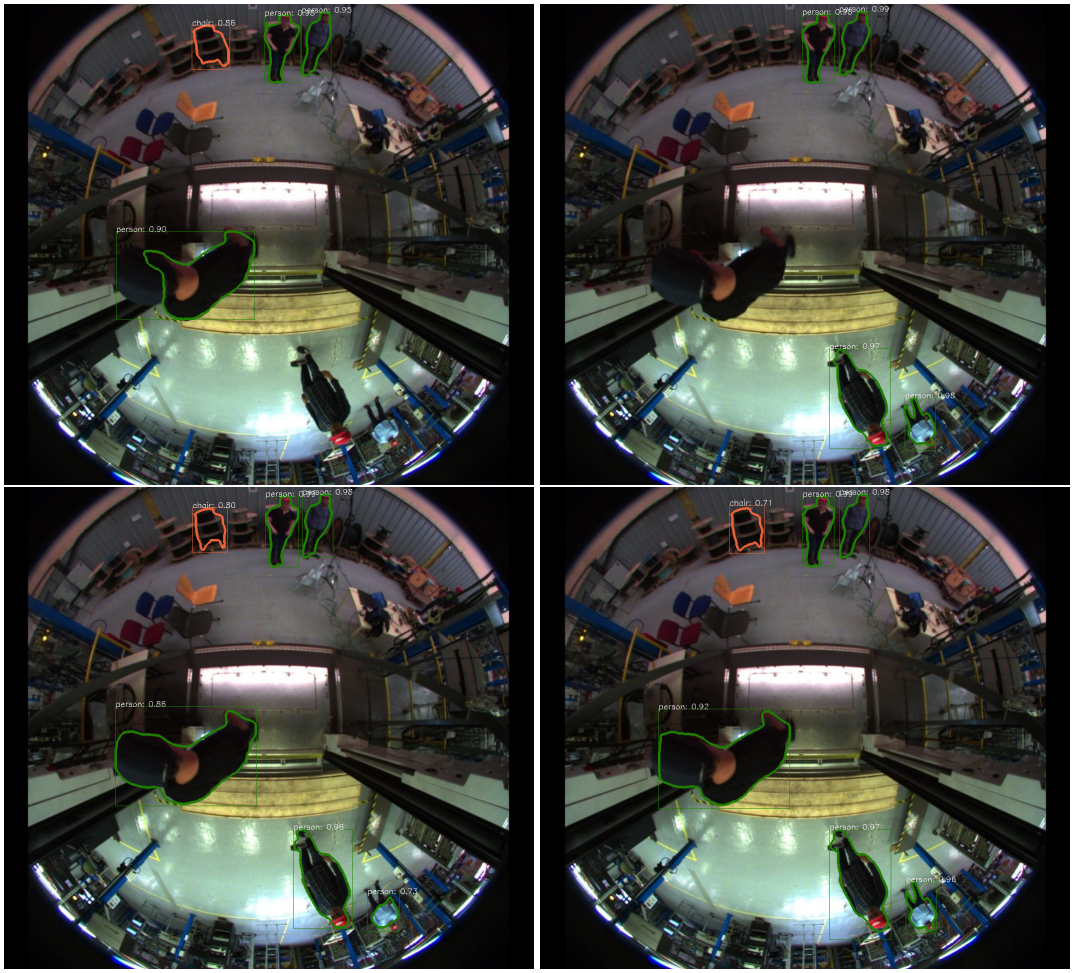


FIGURE 3.13 – Inférence sur une image du dataset trainDoorAug. En haut à gauche : résultat obtenu avec la référence1 (Mask R-CNN ResNet-50 pré-entraîné sur MS COCO). En haut à droite : résultat obtenu avec la référence2 (Mask R-CNN ResNet-50 entraîné avec l'augmentation halfv-flip). En bas à gauche : résultat obtenu Mask R-CNN ResNet-50 entraîné avec une augmentation de 35FE transformations. En bas à droite : résultat avec ResNet-50 Mask R-CNN avec augmentation de 8FE transformations.

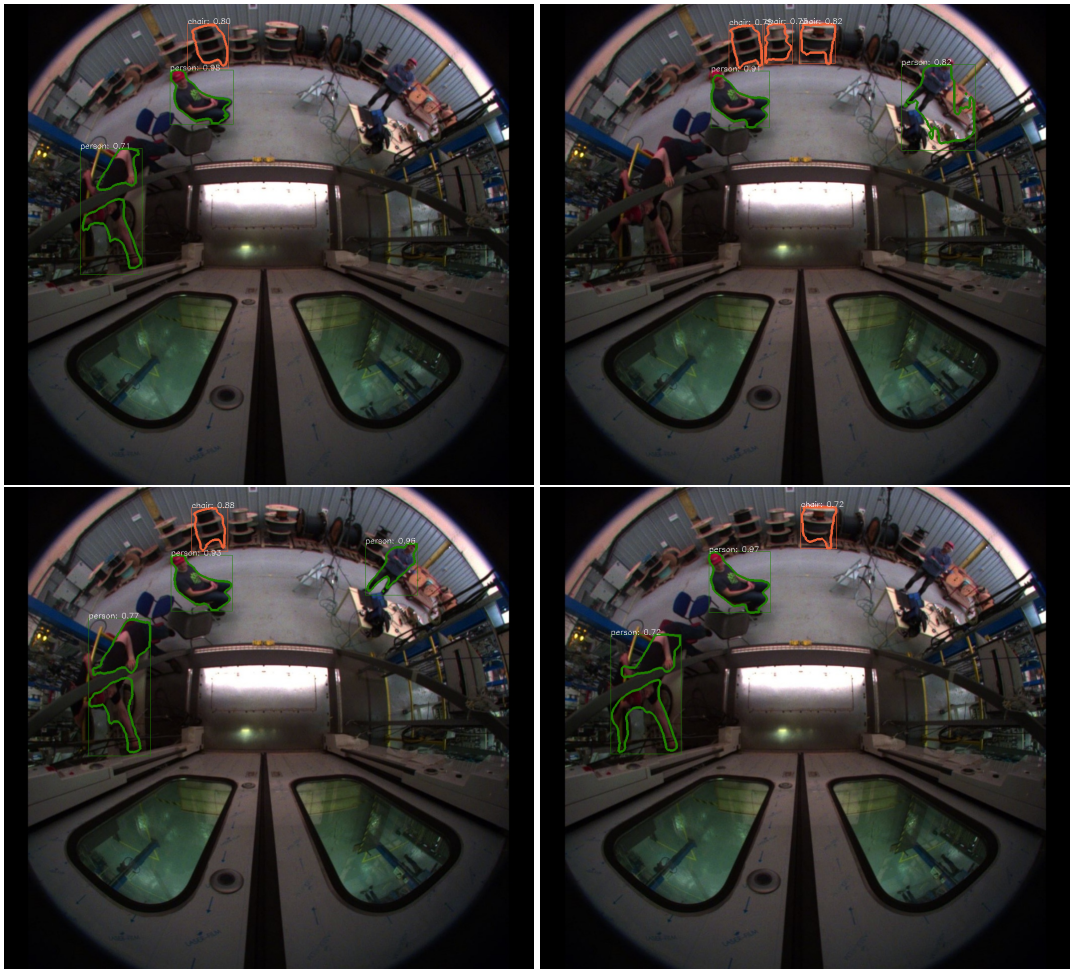


FIGURE 3.14 – Inférence sur une image du dataset trainDoorAug. En haut à gauche : résultat obtenu avec la référence1 (Mask R-CNN ResNet-50 pré-entraîné sur MS COCO). En haut à droite : résultat obtenu avec la référence2 (Mask R-CNN ResNet-50 entraîné avec l'augmentation halfv-flip). En bas à gauche : résultat obtenu Mask R-CNN ResNet-50 entraîné avec une augmentation de 35FE transformations. En bas à droite : résultat avec ResNet-50 Mask R-CNN avec augmentation de 8FE transformations.

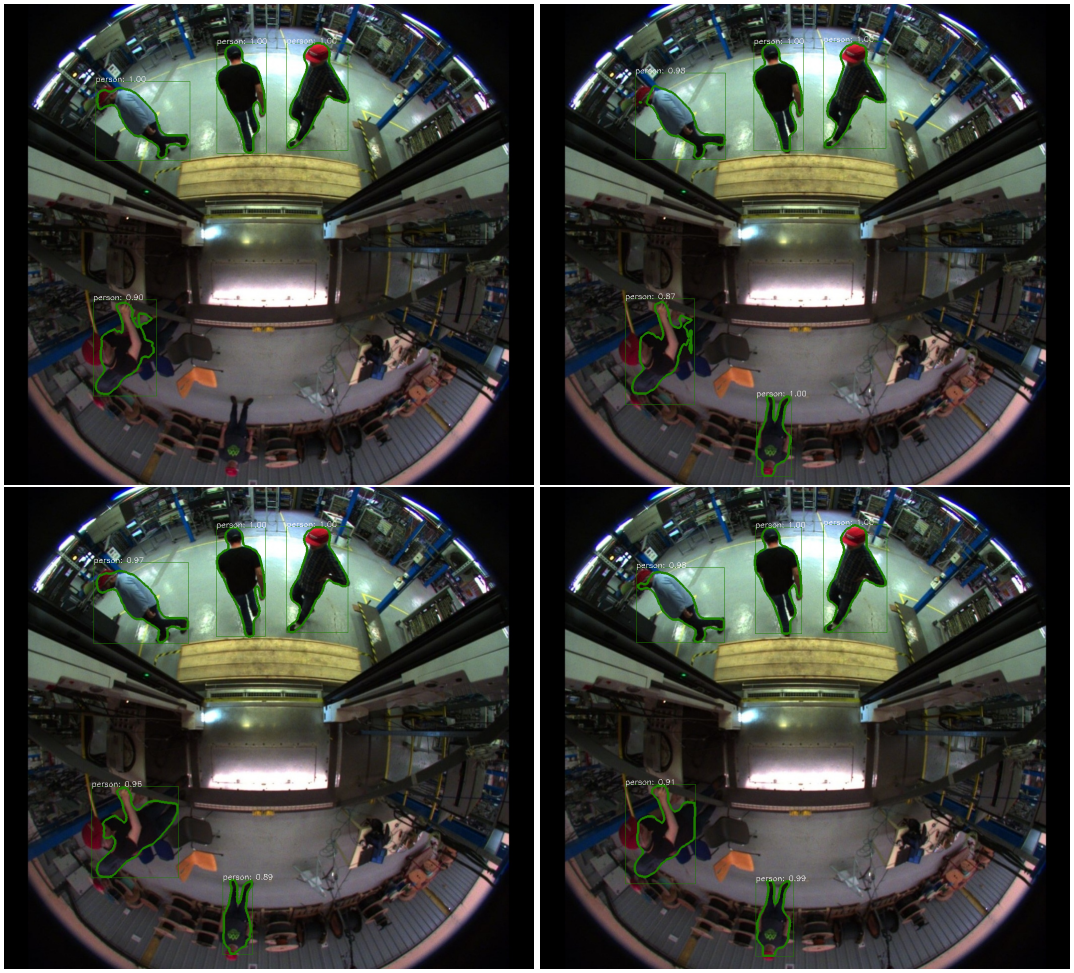


FIGURE 3.15 – Inférence sur une image du dataset trainDoorAug. En haut à gauche : résultat obtenu avec la référence1 (Mask R-CNN ResNet-50 pré-entraîné sur MS COCO). En haut à droite : résultat obtenu avec la référence2 (Mask R-CNN ResNet-50 entraîné avec l'augmentation halfv-flip). En bas à gauche : résultat obtenu Mask R-CNN ResNet-50 entraîné avec une augmentation de 35FE transformations. En bas à droite : résultat avec ResNet-50 Mask R-CNN avec augmentation de 8FE transformations.

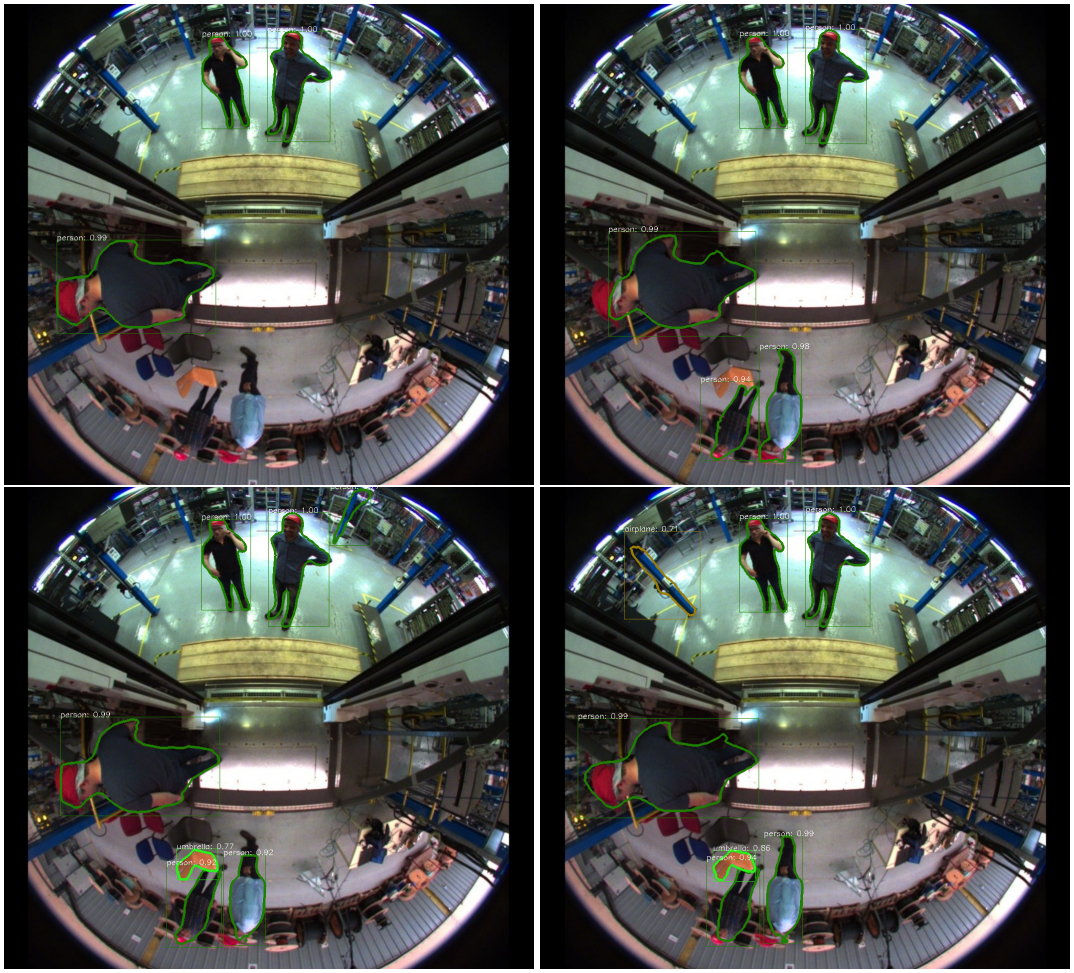


FIGURE 3.16 – Inférence sur une image du dataset trainDoorAug. En haut à gauche : résultat obtenu avec la référence1 (Mask R-CNN ResNet-50 pré-entraîné sur MS COCO). En haut à droite : résultat obtenu avec la référence2 (Mask R-CNN ResNet-50 entraîné avec l'augmentation halfv-flip). En bas à gauche : résultat obtenu Mask R-CNN ResNet-50 entraîné avec une augmentation de 35FE transformations. En bas à droite : résultat avec ResNet-50 Mask R-CNN avec augmentation de 8FE transformations.

3.5 Conclusion

Dans ce chapitre, nous avons proposé une méthode pour affiner un réseau de neurones convolutif conçu pour la segmentation d'instance, afin qu'il soit capable de traiter des images fisheye, sans pour autant dégrader ses performances sur des images rectilinéaires. Les datasets de segmentation d'instance existants à l'heure actuelle manquent d'image fisheye pour permettre l'entraînement de réseaux de neurones profonds. Nous avons alors proposé une méthode d'augmentation de données FE pour générer des images fisheye synthétiques, avec un ratio d'augmentation de 50% permettant de préserver les performances sur les images rectilinéaires. Deux datasets d'évaluation ont été annotés en segmentation d'instance de personnes pour s'assurer du gain de performance obtenu sur des images fisheye réelles dédiées spécifiquement à notre application dans le cadre du programme Train Autonome. De plus, différents ensembles de transformation fisheye sont comparés. Nous concluons qu'un ensemble de 8 transformations FE simples suffit pour obtenir des performances équivalentes voire meilleures sur des images fisheye réelles, que celles obtenues avec 35 transformations sophistiquées. La méthode que nous proposons n'est à priori pas spécifique à la tâche de segmentation d'instance et pourrait être appliquée au défi de la détection de parties du corps dans des vidéos, dont nous parlons dans le chapitre suivant.

Le travail proposé dans ce chapitre a fait l'objet de deux publications [Dufour *et al.*, 2020, Dufour *et al.*, 2021]

Chapitre 4

Réseaux "Space-Time Memory" pour la détection de points clés de personnes dans des vidéos

Dans ce chapitre, nous présentons nos travaux visant à adapter un algorithme de VOS (Video Object Segmentation) à la tâche de détection de points clés de personnes dans des vidéos. Nous expliquons tout d'abord notre motivation, avant de présenter un état de l'art des méthodes pertinentes pour notre problématique. Ensuite, nous détaillons notre méthode, et présentons les différentes expériences qui prouvent la validité de notre concept, avant de détailler les résultats d'une étude ablative montrant l'intérêt de différentes options d'entraînement.

4.1 Motivation

Pour assurer la surveillance et la sécurité dans le futur train autonome, il n'est pas suffisant de détecter la présence des usagers. Il est également indispensable de classifier leur comportement. De grands progrès ayant été réalisés récemment en reconnaissance d'actions à partir de l'étude du squelette (Skeleton-based action recognition), disposer d'une estimation de la pose des personnes devient crucial.

La mise à disposition de nouvelles bases de données et les avancées récentes en apprentissage automatique et en réseaux convolutifs ont mené à de multiples algorithmes de suivi de pose efficaces, que nous passerons en revue dans la section suivante.

Mais avant cela, rappelons que la majorité des algorithmes de suivi de pose actuels utilisent une méthode d'estimation de pose et font ensuite correspondre les squelettes obtenus entre les frames. Certains effectuent cette tâche de manière offline, c'est à dire que la vidéo est traitée comme un tout plutôt que frame par frame, ce qui est inadéquat pour des tâches de surveillance en temps réel. Les méthodes online utilisent généralement un matching frame par frame et ne gardent pas une mémoire long terme

des frames et des prédictions passées, ce qui est gênant pour suivre la trajectoire des personnes sur le long terme. Cependant, des algorithmes de suivi d'objets qui disposent d'une mémoire à long terme de taille arbitraire (dans la limite de la mémoire disponible) ont déjà été développés dans le domaine de la segmentation d'objets dans des vidéos (Video Object Segmentation ou VOS). Notre objectif est alors d'adapter une de ces méthodes de suivi d'objets à mémoire à long terme, à la tâche de détection de points clés de squelette de personnes présentes dans des vidéos. De plus, nous mettrons l'accent sur une méthode bottom-up, capable de gérer efficacement un nombre élevé de personnes.

Nous adaptons donc l'architecture Space-Time Memory network (STM), récemment développée pour la tâche de VOS, et qui utilise une mémoire à long terme pour effectuer le suivi d'une segmentation d'objets dans des vidéos. Par rapport aux méthodes en ligne existantes, la méthode que nous proposons bénéficie d'une mémoire de plusieurs frames et prédictions précédentes, qui peut potentiellement être utilisée pour d'autres tâches de suivi multi-objets par simple affinage.

4.2 État de l'art

Dans cette section, nous présentons plusieurs algorithmes et datasets issus de l'état de l'art et pertinents pour notre problématique de points clés de personnes.

4.2.1 Algorithmes

Dans cette sous-section, nous apportons une vue d'ensemble des algorithmes de l'état de l'art les plus pertinents pour le suivi de pose de personnes dans des trains.

Estimation de pose

L'estimation de pose est une tâche qui consiste à détecter, dans une seule et même image, la pose des personnes présentes, c'est à dire la position des points clés du corps de chaque personne dans l'image ainsi que les arêtes reliant ces points clés, comme illustré dans la figure 4.1. Il s'agit d'un défi facile pour un humain mais très difficile pour un ordinateur. Ceci est activement exploré depuis plusieurs décennies et connaît des progrès significatifs depuis que l'usage des réseaux de neurones convolutifs s'est généralisé.

Les premiers travaux qui se sont intéressés à ce problème se basaient sur une stratégie de modélisation 3D de la pose humaine et ne considéraient qu'une seule personne à la fois, ce qui est parfois appelé "single person pose estimation". Rourke et Al. [O'Rourke et Badler, 1980] effectuent un suivi de pose single-person en propageant les contraintes du modèle 3D entre les frames. Hogg et Al. [Hogg, 1983] utilisent un modèle composé de cylindres pour suivre une personne dans une séquence de 45 images. Yamamoto et

FIGURE 4.1 – Estimation de pose [Cao *et al.*, 2019].

Al. [Yamamoto et Koshikawa, 1991] utilisent un modèle de type "bras robotique" pour suivre une personne dans une séquence vidéo.

A partir de 1994, des travaux ont commencé à utiliser des modèles 2D pour éviter les problèmes liés à la 3D. Baumberg et Hogg [Baumberg et Hogg, 1994] entraînent un modèle de forme flexible sur un grand dataset annoté sans intervention humaine. Morris et Al. [Morris et Rehg, 1998] décrivent un Scaled Prismatic Model (SPM) pour la mise en correspondance 2D afin d'éviter les singularités dues à des mouvements 3D dans l'axe du champ de la caméra. Ioffe et Forsyth [Ioffe et Forsyth, 2001] utilisent un modèle de type mixture-of-tree couplé à un modèle de mouvement pour suivre des personnes dans des séquences vidéos. Mori et Malik [Mori et Malik, 2002] enregistrent des exemples 2D du corps humain pour sélectionner ceux qui correspondent le mieux à la forme de test, puis utilisent les positions de parties du corps pour générer une estimation de la pose en 3D. Ren et Al. [Ren *et al.*, 2005] proposent une méthode qui détecte les parties du corps par la présence de deux lignes parallèles, puis utilisent des contraintes sur la configuration du corps pour obtenir la pose complète. En 2009, Andriluka et Al. [Andriluka *et al.*, 2009] ont adapté le framework de "pictorial structures" [Felzenszwalb et Huttenlocher, 2005], où les poses humaines sont représentées par une collection de parties liées par des arêtes déformables (à la manière d'un ressort), à la détection de personnes et l'estimation de pose, et ont obtenu ainsi les meilleures performances de l'époque sur trois datasets. Ce type de deformable part models [Felzenszwalb *et al.*, 2010] devient donc populaire pendant quelques années [Pishchulin *et al.*, 2013, Yang et Ramanan, 2013], avant l'apparition

des méthodes de deep learning.

DeepPose [Toshev et Szegedy, 2014] fut la première architecture DNN (Deep Neural Networks) pour l'estimation de pose, la formulant en un problème de régression pour trouver les articulations du corps. Les architectures DNN suivantes ont ensuite divergé dans de multiples directions [Fieraru *et al.*, 2018, Wei *et al.*, 2016, Newell *et al.*, 2016, Carreira *et al.*, 2016, Belagiannis et Zisserman, 2017, Sun *et al.*, 2019].

[Fieraru *et al.*, 2018] proposent une méthode pour raffiner une prédiction de pose précédemment générée afin d'améliorer sa qualité. Un CNN d'architecture pré-existante est entraîné pour produire des cartes de confiance pour chaque partie du corps et des cartes indiquant l'offset par rapport au centre des points clés. Des données d'entraînement synthétiques sont créées à partir des annotations de réalité terrain. Les erreurs de prédiction les plus communes sont simulées afin que le réseau apprenne à les corriger.

[Wei *et al.*, 2016] proposent un CNN qui raffine pendant 7 étapes ses prédictions des cartes de confiance des parties du corps. Le modèle est entraîné avec supervision intermédiaire pour chacune des 7 étapes. Ce traitement des prédictions permet de prendre en compte les dépendances à grande échelle entre les parties du corps.

[Newell *et al.*, 2016] proposent une architecture de CNN qui effectue plusieurs étapes de downsampling et upsampling qui forment chacune un sablier, donnant son nom "stacked hourglass". Le réseau est entraîné avec supervision intermédiaire à la sortie de chaque sablier et des connexions sautées relient l'entrée et la sortie des sabliers.

[Carreira *et al.*, 2016] proposent d'entraîner un réseau convolutif afin qu'il prédise l'erreur de la prédiction précédente, selon un processus appelé Iterative Error Feedback (IEF) durant lequel les prédictions sont itérativement corrigées, la correction étant bornée par un maximum pendant l'entraînement.

[Belagiannis et Zisserman, 2017] proposent une architecture en deux modules, l'un feedforward, l'autre récurrent. Le module récurrent se place après le module feedforward, et peut être utilisé plusieurs fois pour raffiner la prédiction, d'une manière inspirée par [Carreira *et al.*, 2016].

[Sun *et al.*, 2019] proposent une nouvelle architecture CNN pour l'estimation de pose nommée HRNet. Alors que la plupart des architectures existantes transforment tout d'abord les représentations de haute à faible résolution, puis retrouvent progressivement des représentations à haute résolution pour la génération des heatmaps, Sun et Al choisissent de conserver des représentations qui gardent la résolution initiale. Ceci permet d'obtenir des heatmaps plus précises.

Estimation de pose multi-personnes

Ensuite, vient une tâche plus difficile qui est l'estimation de pose multi-personnes, où le nombre de personnes présentes dans l'image est inconnu. Les méthodes qui tentent

de résoudre ce problème peuvent être classifiées en top-down et bottom-up. Les premières fonctionnent tout d'abord en détectant des objets de grande taille (les humains ici). Elles prédisent ensuite l'emplacement d'objets de plus petite taille comme les points clés des squelettes. Cela implique l'usage de deux algorithmes : l'un pour la détection de personne, et l'autre pour estimer une pose humaine à partir d'une fenêtre de personne particulière. Les méthodes bottom-up commencent elles par traiter les objets de petite taille avant de travailler sur des échelles plus grandes. Dans le cas de l'estimation de pose, les points clés de squelettes sont détectés en premier, et sont ensuite assemblés en squelettes complets ou partiels. Les méthodes bottom-up ont généralement des propriétés de mise à l'échelle favorables en terme de rapidité de traitement d'un grand nombre de personnes, ce qui les rend plus adaptées pour des tâches de surveillance.

Un DNN produisant des cartes de confiance en la présence de points clés du corps, combiné avec une Non-Maximum Suppression (NMS), est une méthode qui a fait ses preuves pour construire des méthodes d'estimation de pose bottom-up. Cao et al. [Cao *et al.*, 2019], inspirés par [Wei *et al.*, 2016] et [Insafutdinov *et al.*, 2016] ont créé un DNN multi-stage de raffinement, entraîné avec une supervision intermédiaire, qui produit des cartes de confiance en la présence de parties du corps (keypoints confidence maps), et des Part Affinity Fields (PAF). Le modèle prend en entrée une image RGB qui est passée dans un CNN constitué des 10 premières couches affinées de VGG-19 [Simonyan et Zisserman, 2015], ce qui produit un ensemble de feature maps F. Ensuite, les premiers étages (qui vérifient, pour l'étage t, $t \leqslant T_P$) prédisent des ensembles de PAF (T_P étant le nombre d'étages qui prédisent des PAF), qui sont des champs vectoriels indiquant à la fois la probabilité de présence et la direction d'un membre qui relie deux articulations du corps. Les derniers étages (qui vérifient, pour l'étage t, $T_P < t \leqslant T_P + T_C$) produisent des cartes de confiance pour la présence de points clés du corps (T_C étant le nombre d'étages qui prédisent des points-clés du corps). On a donc T étages avec $t \in 1, ..., T$. Ce fonctionnement est illustré dans la figure 4.2. L'ensemble $S = (S_1, S_2, ..., S_J)$ contient J cartes de confiance, une par point clés du corps. L'ensemble $(L_1, L_2, ..., L_C)$ contient C champs vectoriels, un par arête entre deux points clés du corps, avec $L_c \in \mathbb{R}^{w \times h \times 2}, c \in \{1, 2, ..., C\}$ où C est le nombre d'arêtes considérées. Cette architecture a été choisie car les auteurs ont expérimentalement montré que la prédiction des cartes de confiance bénéficie de la connaissance préalable des PAF, mais que l'inverse n'est pas vrai. A la fin de chaque étage, une loss est calculée pour effectuer une supervision intermédiaire inspirée de [Wei *et al.*, 2016], qui permet d'éviter le problème de disparition du gradient. Spécifiquement, la fonction de loss de la branche PAF à l'étage t_i et la fonction de loss de la branche points clés à l'étage t_k sont données par les équations 4.1 et 4.2 suivantes :

$$f_L^{t_i} = \sum_{c=1}^{C} \sum_{p} W(p) \cdot \| L_c^{t_i}(p) - L_c^*(p) \|_2^2 \tag{4.1}$$

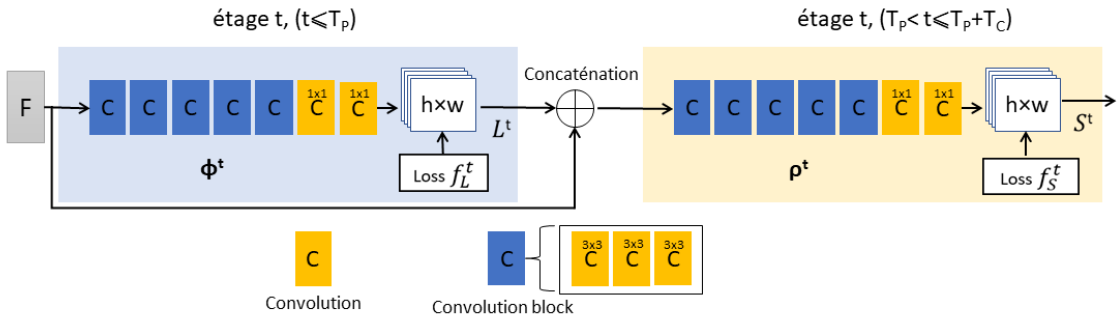


FIGURE 4.2 – Architecture du modèle multi-étages OpenPose [Cao *et al.*, 2019].

$$f_S^{t_k} = \sum_{j=1}^{J} \sum_{p} W(p) \cdot \|S_j^{t_k}(P) - S_j^*(p)\|_2^2 \tag{4.2}$$

La fonction de loss totale est donnée par la formule :

$$f = \sum_{t=1}^{T_P} f_L^t + \sum_{t=T_P+1}^{T_P+T_C} f_S^t \tag{4.3}$$

Plus spécifiquement, le premier étage de l'architecture produit un ensemble de PAF $L^1 = \phi^1(F)$ où ϕ^1 fait référence à l'inférence du CNN à l'étage 1. Dans chaque étage suivant, les prédictions de l'étage précédent et les feature maps de l'image F sont concaténées et utilisées pour produire des prédictions raffinées.

$$L^t = \phi^t(F, L^{t-1}), 2 \leqslant t \leqslant T_P \tag{4.4}$$

où ϕ^t est le CNN à l'étage t, et T_P le nombre total d'étages PAF. Après T_P itérations, le processus est répété pour la production des cartes de confiance des parties du corps, à partir de la prédiction de PAF la plus récente.

$$S^{T_P} = \rho^t(F, L^{T_P}), t = T_P \tag{4.5}$$

$$S^t = \rho^t(F, L^{T_P}, S^{t-1}), T_P < t < T_P + T_C \tag{4.6}$$

où ρ^t fait référence au CNN à l'étage t, et T_C au nombre total d'étages pour les cartes de confiance.

Après que les cartes de confiance en la présence de points clés du corps aient été produites à la dernière inférence, une étape de NMS (Non-Maximum Suppression) permet d'obtenir les candidats de points clés du corps. Pour calculer l'affinité entre des points clés liés (par exemple entre l'épaule gauche et le coude gauche), le PAF correspondant est échantillonné régulièrement le long de l'arête entre les points clés. Grâce aux PAF, le calcul de l'association entre les points clés permet d'utiliser ensuite un simple

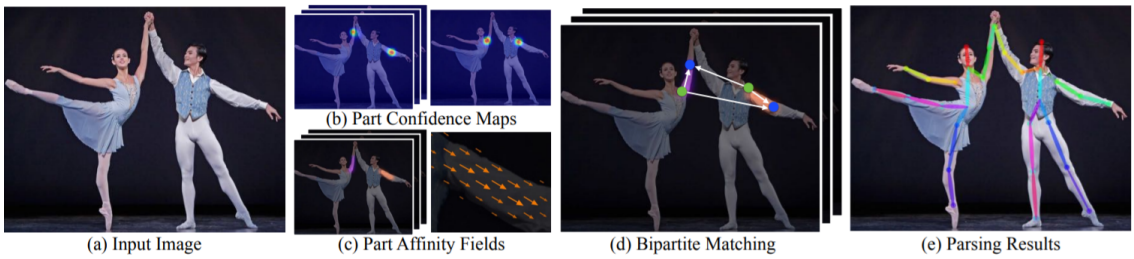


FIGURE 4.3 – Pipeline de la méthode OpenPose [Cao *et al.*, 2019]. (a) l'image entière est fournie au modèle pour prédire à la fois (b) les cartes de confiance pour la présence de points clés du corps et (c) les PAF pour l'association de points clés. (d) un matching biparti est réalisé pour associer les points clés candidats à l'aide des PAF. (e) les points clés sont associés en pose complète pour chaque personne présente dans l'image.

algorithme de matching de graphe biparti, suivant l'algorithme hongrois, pour associer les points clés entre eux et ainsi former des squelettes complets. Le pipeline complet d'OpenPose est illustré dans la figure 4.3. Cet algorithme a inspiré notre utilisation de cartes de confiance avec NMS ainsi que l'idée de raffinage de la prédiction grâce à des inférences successives prenant en entrée, la sortie de l'inférence précédente.

Plus récemment, [Kreiss *et al.*, 2019] utilisent des Part Intensity Fields (PIF) au lieu des part confidence maps, ainsi que des PAF composites plus complexes pour améliorer la précision et la robustesse à différentes échelles d'instances de personnes.

Pose tracking

Motivé par les nombreuses applications en reconnaissance d'actions et analyse de scènes, de grands progrès ont été réalisé en suivi et estimation de pose dans des vidéos depuis quelques années. Ce phénomène a aussi été rendu possible par la publication de datasets de taille suffisante pour entraîner des modèles basés sur l'apprentissage profond.

Les algorithmes de suivi de pose peuvent généralement être classifiés en online et offline. Les algorithmes online traitent les frames d'une séquence vidéo de manière séquentielle et peuvent fonctionner pendant un temps indéterminé, tandis que les algorithmes offline [Insafutdinov *et al.*, 2017, Iqbal *et al.*, 2017] prennent en entrée une séquence vidéo entière. Dans cette revue de l'état de l'art, nous allons nous concentrer sur les algorithmes online, qui sont les plus applicables aux tâches de surveillance vidéos. Similairement aux estimateurs de pose single frame, ils peuvent être classifiés en top-down [Girdhar *et al.*, 2018, Xiao *et al.*, 2018, Xiu *et al.*, 2018, Snower *et al.*, 2020, Ning et Huang, 2020] et bottom-up [Doering *et al.*, 2018, Jin *et al.*, 2019, Raaj *et al.*, 2019].

Les algorithmes top-down utilisent un détecteur de personnes pour obtenir des boîtes englobantes dans lesquelles les poses sont estimées, avant de les suivre dans le temps. Ils

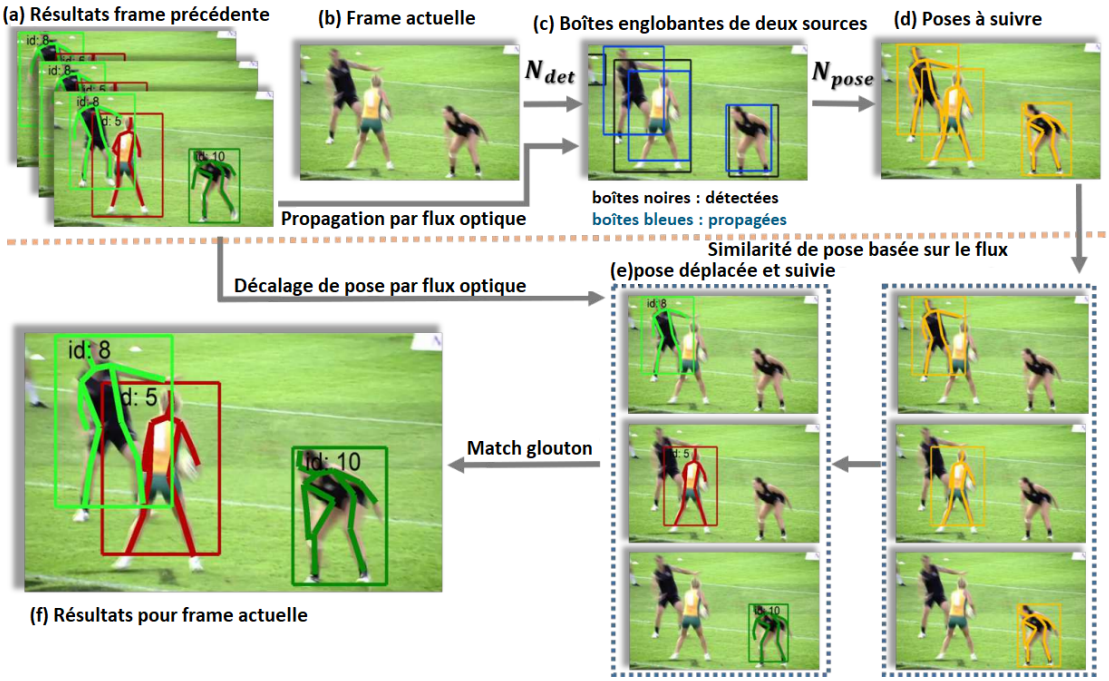


FIGURE 4.4 – Algorithme d'une baseline simple pour le suivi de poses [Xiao *et al.*, 2018].

se différencient principalement par les estimateurs choisis, et la manière dont ils sont intégrés ensemble. Ils partagent les caractéristiques d'être composés de multiples sous-systèmes et d'avoir des temps de calcul qui dépendent du nombre de personnes dans la vidéo. Les approches bottom-up produisent des cartes de confiance pour détecter chaque parties du corps, et ensuite groupent les points clés au sein de chaque frame (pour former des squelettes de personnes) et dans le temps.

[Girdhar *et al.*, 2018] proposent une approche top-down en deux étapes. Premièrement, une version 3D de Mask R-CNN prend des clips vidéos courts en entrée et génère des tubes de personnes avec les points clés associés. Le suivi dans le temps est formulé comme un problème de matching de graphe biparti entre chaque paire de frames, qui est résolu avec l'algorithme hongrois. L'affinité entre des paires successives de boîtes englobante est calculée à partir de la superposition des boîtes englobantes.

[Xiao *et al.*, 2018] proposent une baseline simple pour l'estimation et le suivi de poses dans des vidéos. Un réseau convolutif résiduel avec tête de déconvolution génère des heatmaps pour chaque parties du corps. Le flux optique est utilisé pour propager les boîtes englobantes de la frame précédente afin de combler les lacunes du détecteur. Enfin, un matching glouton est réalisé à partir d'une mesure de similarité basée sur le flux optique et permet d'associer les points clés dans le temps. L'algorithme complet est illustré dans la figure 4.4.

[Xiu *et al.*, 2018] utilisent l'estimateur de pose top-down multi-person RMPE (Regio-

nal Multi-Person Pose Estimation) [Fang *et al.*, 2017], qui utilise d'une part Faster R-CNN [Ren *et al.*, 2015] pour la détection des boîtes englobantes de personnes et d'autre part Hourglass Network avec PRM (Pyramid Residual Module) [Yang *et al.*, 2017] comme estimateur de pose single-person. Les poses sont ensuite associées selon la dimension temporelle pendant une phase appelée "Pose Flow Building" avant d'être passées au crible pendant l'étape "Pose Flow NMS" qui fusionne les tracklets qui sont trop proches les uns des autres.

[Snower *et al.*, 2020] utilisent HRNet [Sun *et al.*, 2019], une approche top-down d'estimation de pose, avant de raffiner les estimations grâce à leurs poses associées dans la frame précédente. Les poses sont associées temporellement à l'aide d'une technique provenant du NLP (Natural Language Processing) en étant d'abord transformées en tokens puis traitées par un transformer qui produit des scores de matching.

[Ning et Huang, 2020] proposent une méthode top-down, online qui utilise un détecteur de personnes pour obtenir des boîtes englobantes de personnes, avant d'utiliser un estimateur de pose single-person sur chaque boîte englobante. La propagation dans le temps est réalisée avec un réseau siamois qui effectue des convolutions de graphe sur les poses détectées. Il est entraîné par une contrastive loss pour générer des vecteurs de caractéristiques qui permettent le matching ou non des deux poses reçues en entrée.

[Doering *et al.*, 2018] utilisent un réseau siamois qui produit, à partir des caractéristiques obtenues par un backbone, des cartes de confiance des parties du corps et des PAF à la manière d'OpenPose [Cao *et al.*, 2019] pour prédire les poses dans les frames t et t-1 pendant l'inférence spatiale. Ils effectuent ensuite un matching temporel à l'aide d'un CNN temporel qui produit des champs vectoriels indiquant la direction du mouvement des parties du corps.

[Jin *et al.*, 2019] proposent une méthode constituée d'un module "SpatialNet" et d'un module "TemporalNet". SpatialNet effectue à partir d'une frame, la prédiction de heatmaps pour chaque points clé du corps : des Keypoints Embeddings (KE), des Spatial Instance Embeddings (SIE) et des cartes geometric-ordinal qui sont auxiliaires et ne servent qu'à faciliter l'apprentissage des KE. SpatialNet permet la génération de propositions de poses. TemporalNet se base sur les résultats du SpatialNet pour générer des Human Embeddings (HE) et Temporal Instance Embeddings (TIE) qui permettent le groupement des poses dans la dimension temporelle.

Raaj and al. [Raaj *et al.*, 2019] ont été également inspiré par [Cao *et al.*, 2019] et ont étendu l'idée des Part Affinity Fields (PAF) pour l'appliquer à la dimension temporelle, effectuant le suivi de pose en matchant les points clés de frame à frame à l'aide des TAF (Temporal Affinity Fields). L'algorithme est nommé STAF (Spatio-Temporal Affinity Fields). A chaque frame, le réseau extrait des caractéristiques de l'image grâce à un backbone VGG. Pour la frame I^t à l'instant t dans la vidéo, les caractéristiques sont calculées ainsi : $V^t = \phi_V(I^t)$. Ensuite, les PAF, cartes de confiance de keypoints et TAF sont cal-

culées récursivement selon les équations 4.7 à 4.9 ci-dessous :

$$L^t = \phi_L(V^t, L^{t-1}) \tag{4.7}$$

$$K^t = \phi_K(V^t, L^t, K^{t-1}) \tag{4.8}$$

$$R^t = \phi_R(V^{t-1}, Vt, L^{t-1}, L^t, R^{t-1}) \tag{4.9}$$

V, L, K et R étant respectivement les sorties du backbone VGG, des PAF, keypoints, et TAF. Similairement à [Cao *et al.*, 2019], une étape de NMS (Non Maximum Suppression) est appliquée sur la carte de confiance des keypoints, ce qui permet d'obtenir des keypoints candidats. Les PAF sont ensuite échantillonnés sur les lignes entre les keypoints pour obtenir le score de matching entre toutes les paires de keypoints. Cela permet d'ordonner les matchs par leur score. Ensuite, pour chaque match : (i) une nouvelle pose est initialisée si les deux keypoints ne sont pas encore assignés, (ii) si l'un des keypoints est assigné, l'autre est assigné à la pose existante, (iii) si les deux sont déjà assignés à la même pose, le score de matching est mis à jour, (iv) si les deux sont assignés à des poses différentes, elles sont fusionnées. Enfin, des id sont assignés à chaque pose en fonction de l'id le plus commun pour les keypoints de la pose dans la frame précédente. Le fonctionnement de STAF est illustré dans la figure 4.5. STAF traite les séquences vidéos de manière récurrente en générant des points clés et des connexions entre points clés dans chaque frame grâce aux Part Affinity Fields (PAF), et des connexions entre points clés à travers le temps grâce aux Temporal Affinity Fields (TAF). Le tout s'appelle Recurrent Spatio-Temporal Affinity Fields (STAF). Chaque module prend en entrée des sorties provenant d'autres modules à la fois dans la frame actuelle et dans la frame précédente et raffine l'estimation. Durant l'inférence, le réseau opère sur une seule frame vidéo à chaque pas temporel en utilisant les informations des frames passées. Les cartes de confiance prédites sont ensuite utilisées pour détecter et suivre les personnes. Les points clés sont estimés en premier, puis associés en pose dans l'espace grâce aux PAF, puis dans le temps grâce aux TAF et aux tracklets des frames précédentes. Le modèle est pré-entraîné en Mode Image, pendant lequel des images individuelles sont données et les prédictions de PAF, TAF et keypoints sont donc raffinées pour une seule image. L'entraînement principal est ensuite réalisé en Mode Video où le modèle est exposé à des séquences vidéos. Le pré-entraînement et l'entraînement sont réalisés avec les bases de données MS-COCO, MPII et Posetrack. Pendant l'entraînement en Mode Vidéo, des mouvements sont synthétisés avec des mises à l'échelle, des rotations et des translations pour les datasets d'images MS-COCO et MPII. La méthode que nous proposons pour la détection de points clés de squelette est similaire à STAF pour l'entraînement et pour l'usage de cartes de confiance de keypoints, mais diffère en d'autres points car notre modèle dispose d'une mémoire à long terme, et ne vise qu'à remplacer la partie de STAF qui effectue la détection de points clés de squelette.

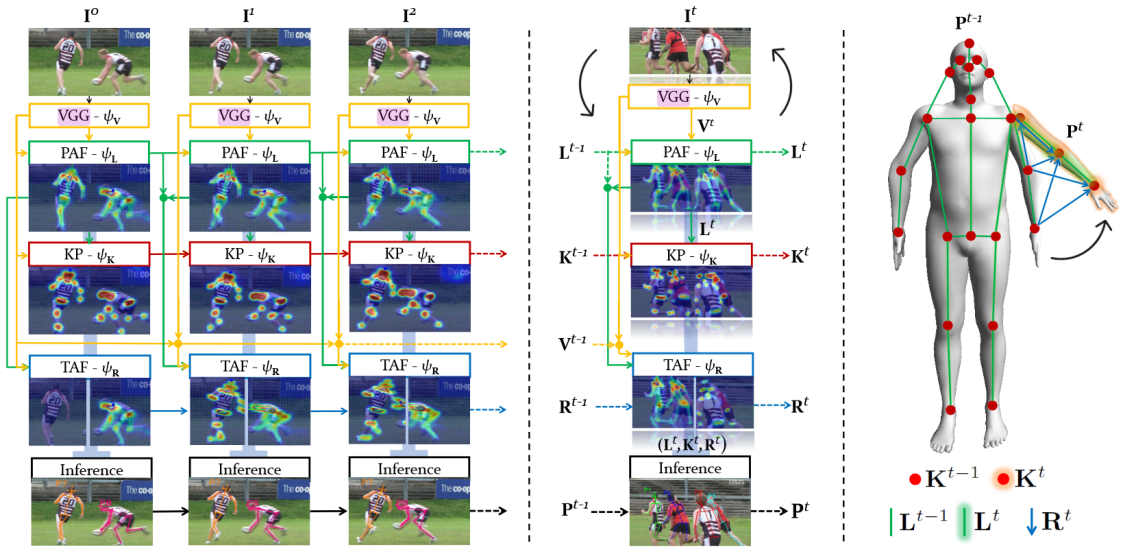


FIGURE 4.5 – Fonctionnement de l'algorithme STAF [Raaj *et al.*, 2019] **Gauche** : fonctionnement de STAF. L'algorithme prédit des points clés et des connexions entre points clés dans l'espace grâce aux Part Affinity Fields (PAF), et des connexions à travers le temps grâce aux Temporal Affinity Fields (TAF). **Centre** : durant l'inférence, le réseau traite chaque frame une à une en utilisant les informations des frames passées. **Droite** : les cartes de confiance prédites sont utilisées pour détecter et suivre les personnes. Les points clés (rouge) sont estimés en premier, puis associés en pose grâce aux PAF (vert), TAF (bleu), et aux tracklets des frames précédentes.

Doering and al. [Doering *et al.*, 2018] ont suivi une inspiration similaire pour créer un réseau siamois encodant deux frames consécutives pour obtenir des cartes de confiance, des PAF et des Temporal Flow Fields (TFF) pour suivre les points clés de frame à frame.

Jin and al. [Jin *et al.*, 2019] ont utilisé un SpatialNet pour produire des heatmaps de points clés du corps, des Keypoint Embeddings (KE) et des instance embeddings (SIE) pour grouper les propositions de points clés en propositions de poses humaines, avant d'utiliser un TemporalNet pour effectuer le suivi dans le temps.

Video Object Segmentation

La Video Object Segmentation (VOS) [Pont-Tuset *et al.*, 2017] est une tâche dans laquelle un ou plusieurs objets doivent être segmentés dans des vidéos données en entrée.

La segmentation d'objets dans des vidéos peut être non-supervisée (unsupervised) ou semi-supervisée (semi-supervised). Le défi de VOS non-supervisée implique que l'objet à segmenter n'est pas donné à l'algorithme. Il s'agit donc d'utiliser la salience visuelle ou le flux optique pour déterminer l'objet à segmenter. Le défi de VOS semi-supervisée implique une annotation de vérité terrain pour la première frame. La VOS non-supervisée est un défi très spécifique, que nous ne considérons pas pertinent pour le suivi de pose. Nous nous concentrons donc dans la suite de ce mémoire sur les techniques de VOS semi-

supervisées et le terme VOS fait référence à ce défi.

Avant la popularisation des réseaux convolutifs profonds la VOS était approchée comme un problème d'optimisation de graphe spatio-temporel. Pour réduire la complexité, la plupart des méthodes groupent les pixels de différentes manières. [Chang *et al.*, 2013, Grundmann *et al.*, 2010] utilisent une sursegmentation en superpixels. [Ramakanth et Babu, 2014, Fan *et al.*, 2015] divisent l'image en patches. [Märki *et al.*, 2016] proposent d'utiliser une grille bilatérale, un espace de dimension supérieure, où chaque pixel est transporté en fonction de sa couleur et de ses coordonnées spatiales et temporelles. Ceci permet d'effectuer un graph cut efficace dans cet espace afin de classifier chaque pixel en foreground ou background. D'autre part, l'optimisation peut considérer la séquence vidéo complète [Märki *et al.*, 2016], un sous-ensemble de frames [Grundmann *et al.*, 2010] ou bien seulement les résultats à la frame n pour obtenir le masque à la frame n+1 [Chang *et al.*, 2013, Fan *et al.*, 2015, Ramakanth et Babu, 2014]. Pour une application temps réel, la dernière catégorie est la plus pertinente. Cependant, le flux optique est utilisé par certaines méthodes précédentes [Grundmann *et al.*, 2010, Ramakanth et Babu, 2014], ce qui ajoute un temps de calcul non-négligeable et limite leur utilisation pour une application temps réel. Dans l'ensemble, ces techniques mettent souvent l'accent sur la consistance temporelle des segmentations prédites pour effectuer l'optimisation.

Récemment, le défi de VOS a connu de grands progrès grâce à l'application des méthodes d'apprentissage profond convolutif. Ces méthodes sont entraînées offline [Jampani *et al.*, 2017, Hu *et al.*, 2017, Li et Loy, 2018, Chen *et al.*, 2018] ou online [Perazzi *et al.*, 2017, Caelles *et al.*, 2017]. Un entraînement offline a lieu avant l'utilisation de l'algorithme sur la vidéo, alors qu'un entraînement online est effectué pour une vidéo spécifique. La plupart des algorithmes avec entraînement online effectuent quelques centaines de pas d'entraînement sur la première frame de la vidéo. Les meilleurs résultats sont obtenus avec un entraînement offline, mais cela nécessite un temps de calcul considérable et diminue l'intérêt de ces techniques pour une application temps réel.

On peut également noter deux grandes catégories d'approches pour la VOS : les approches par propagation [Perazzi *et al.*, 2017, Khoreva *et al.*, 2019, Hu *et al.*, 2017, Li et Loy, 2018] et les approches par détection [Caelles *et al.*, 2017, Maninis *et al.*, 2018, Yoon *et al.*, 2017, Bao *et al.*, 2018, Chen *et al.*, 2018, Hu *et al.*, 2018]. Les approches par propagation apprennent un propagateur de masque d'objet et travaillent donc à partir de la frame précédant la frame cible, tandis que les approches par détection utilisent un détecteur d'objet qui prend en entrée la première frame et son masque. D'autres approches essaient de tirer parti à la fois de la propagation et de la détection [Yang *et al.*, 2018, Oh *et al.*, 2018].

[Perazzi *et al.*, 2017] entraînent un réseau convolutif offline, uniquement sur des images statiques, avant de l'affiner online sur la première frame de la vidéo. Le réseau prend en entrée la frame courante et le masque de l'objet dans la frame précédente, et produit

une segmentation pour la frame courante.

[Caelles *et al.*, 2017] proposent un entraînement online pour segmenter chaque frame de la vidéo, une à la fois. Le FCN, pré-entraîné sur ImageNet, se divise en deux branches : l'une prédit le foreground, l'autre prédit les contours des objets et permet d'ajuster la prédiction du foreground aux contours. Les prédictions sont de si bonne qualité que le besoin de consistance temporelle ne se fait plus sentir.

[Jampani *et al.*, 2017] proposent un réseau convolutif qui prend en entrée les masques précédents, projette les pixels dans un espace bilatéral, où une convolution est effectuée, avant de retourner dans l'espace image original, et enfin de passer dans un réseau convolutif spatial qui produit la carte de segmentation d'objets pour la frame actuelle.

[Hu *et al.*, 2017] effectuent la VOS semi-supervisée multi-objets en utilisant un binary segmentation net par instance d'objet, dont les sorties sont ensuite fusionnées pour produire les cartes de segmentation binaires. Le réseau prend en entrée l'image courante ainsi que la magnitude du flux optique et le masque de la frame précédente déformé en fonction du flux optique.

[Li et Loy, 2018] proposent une méthode qui extrait tout d'abord des caractéristiques avec un backbone pré-entraîné, avant de produire des boîtes englobantes candidates avec un RPN (Region Proposal Network), puis utilise deux modules : Re-ID (ré-identification) et Re-MP (Recurrent Mask Propagation). Re-ID produit pour chaque RoI (Region of Interest) un masque binaire et des caractéristiques de masque pour retrouver des objets manquants qui réapparaissent sous différentes poses. Re-MP prend en entrée le masque déformé grâce au flux optique et propage les masques entre frames adjacentes afin de former les tracklets.

[Chen *et al.*, 2018] formulent le problème comme récupération des pixels dans un espace d'embedding appris. Un extracteur de caractéristiques prend en entrée l'image de référence et l'image courante, et une tête d'embedding se charge de placer les points qui appartiennent au même objet proche les uns des autres dans l'espace d'embedding. Ceci permet à un classifieur plus-proche-voisin de classifier chaque pixel en foreground ou background. Cette architecture permet aussi d'obtenir un temps de calcul faible.

[Hu *et al.*, 2018] proposent l'utilisation d'un CNN pour extraire les caractéristiques de foreground et background dans la frame initiale. Cela permet de calculer la similarité avec le foreground et le background pour chaque pixel dans l'image courante. Cet algorithme, illustré dans la 4.6, est également l'un des plus rapides de l'état de l'art.

[Yang *et al.*, 2018] utilisent deux réseaux modulateurs : un visuel et un spatial, pour modifier en temps réel les cartes de caractéristiques d'un réseau de segmentation et ainsi l'adapter à la segmentation d'un objet spécifique. Cela permet d'obtenir des performances similaires à celles qui effectuent un affinage sur la première frame, tout en proposant un temps de calcul faible.

[Oh *et al.*, 2018] utilisent deux extracteurs de caractéristiques avec paramètres parta-

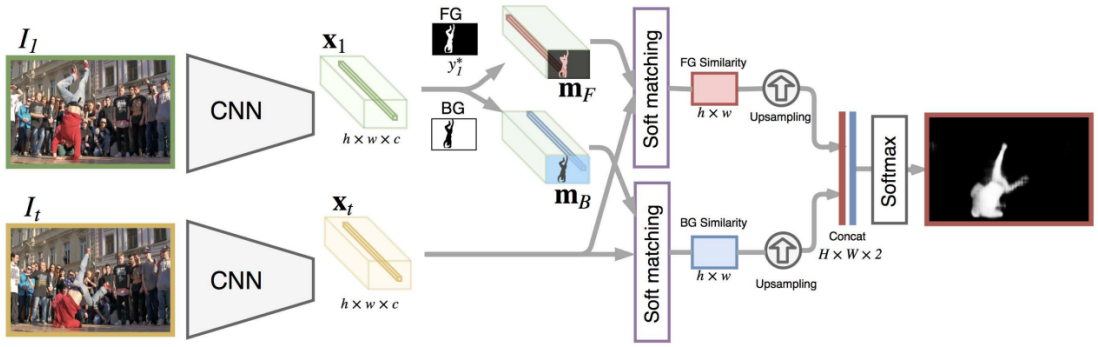


FIGURE 4.6 – Fonctionnement de l'algorithme de VOS videomatch [Hu *et al.*, 2018].

gés (réseaux siamois). Le premier prend en entrée la frame cible et le masque de la frame précédente. Le deuxième prend en entrée la frame de référence et le masque associé. Un décodeur met à l'échelle les caractéristiques jusqu'à obtenir le masque de foreground pour la frame cible. Cette architecture permet des performances parmi les meilleures de l'état de l'art en considérant à la fois la vitesse de traitement et la qualité des résultats. Les auteurs ont poursuivi leurs travaux avec une nouvelle architecture conçue pour la tâche de VOS nommée STM (Space-Time Memory Networks) [Oh *et al.*, 2019]. En utilisant un système flexible basé sur les memory networks [Kumar *et al.*, 2016, Sukhbaatar *et al.*, 2015, Miller *et al.*, 2016] utilisés en NLP (Natural Language Processing), il peut faire usage d'un nombre arbitraire d'images et de prédictions passées pour prédire la segmentation de l'objet dans la frame courante, en étant seulement limité par la mémoire disponible. STM a été pré-entraîné sur des séquences synthétiques créées à partir de plusieurs datasets d'images, puis entraîné sur des datasets de vidéos, tels que Youtube-VOS [Xu *et al.*, 2018] et DAVIS-2017 [Pont-Tuset *et al.*, 2017]. Le modèle a obtenu des résultats équivalents et a établi un nouvel état de l'art, en termes de qualité et de rapidité du suivi. Pour ces raisons auxquelles s'ajoutent son adaptabilité à d'autres tâches que la VOS, nous avons décidé d'adapter ce modèle à la tâche de détection de parties du corps multi-personnes dans des vidéo. De plus son architecture est utilisable pour d'autres tâches que la VOS. Cette contribution est détaillée dans la section 4.3.

4.2.2 Datasets

Entraîner des modèles de suivi de pose de l'état de l'art nécessite des datasets massifs et de haute qualité. Dans cette partie, nous présentons les principaux datasets pertinents en estimation et en suivi de pose.

MPII [Andriluka *et al.*, 2014] est un dataset pour l'estimation de pose sourcé de Youtube, annoté grâce à Amazon Mechanical Turk (AMT) et des annotateurs sélectionnés pour leur qualité, à l'aide d'outils d'annotation adaptés de [Maji, 2011]. 24920 frames

contenant 40522 personnes sont annotées. Les annotations de poses couvrent 15 points clés (nez, oeil gauche, oeil droit, épaule gauche, épaule droite, coude gauche, coude droit, poignet gauche, poignet droit, hanche gauche, hanche droite, genoux gauche, genoux droit, cheville gauche, cheville droite). Les frames sont classées par une hiérarchie en deux niveaux, parmi 823 activités et 21 catégories d'activités. Nous n'utilisons pas ce dataset car sa taille et sa diversité sont inférieures au dataset de référence MS-COCO.

MS-COCO [Lin *et al.*, 2014] est un dataset contenant de nombreuses images très variées, annotées pour la segmentation d'instance (la tâche d'assigner à chaque pixel dans une image une classe d'objet ou à la classe background). En 2017, les auteurs ont publié des annotations de pose de personnes, couvrant 17 points clés (les 15 points clés de MPII, plus l'oreille gauche et l'oreille droite). Les exemples sont séparés en trois sous ensembles "train", "validation" et "test". Nous utilisons ici les deux premiers ensembles.

PoseTrack18 [Andriluka *et al.*, 2018] contient un grand nombre de séquences vidéos courtes, tirées du dataset populaire MPII Human Pose Dataset [Andriluka *et al.*, 2014]. Le dataset est subdivisé en train, validation et test contenant respectivement 550 vidéos, 170 vidéos, et 375 vidéos. Ils ont été annotés avec 15 points clés du corps humain (tête, nez, cou, épaules, coudes, poignets, hanches, genoux, chevilles). Les vidéos présentent des contextes variés (sports, travail, vie de tous les jours, ...), une grande diversité en termes de mouvements et de poses corporelles, ainsi que des caractéristiques qui compliquent le traitement d'image, comme l'occultation des personnes entre elles, des changements rapides d'échelles, des foules denses. La majorité des séquences contiennent entre 41 et 151 frames, ce qui correspond à quelques secondes de temps réel. Dans la suite de ce manuscrit, nous utilisons PoseTrack18_train pour l'affinage de nos modèles, en extrayant de courtes séquences (2 à 5 frames) de vidéos le constituant. Nous utilisons PoseTrack18_Validation pour les évaluer, soit en prenant les vidéos complètes et en essayant de suivre les personnes (ce qui constitue un défi que nous appelons PoseTrack18_Validation FULL), soit en prenant de courtes séquences de 2 à 5 frames (ce qui constitue un défi que nous appelons PoseTrack18_Validation SHORT).

Dans la suite de ce chapitre, les datasets qui sont utilisés pour l'entraînement et l'évaluation des différents modèles sont MS-COCO et Posetrack.

4.3 Méthode

Cette section donne des détails sur l'architecture de réseaux de neurones profond que nous adaptons, puis explique les changements qui ont été effectués pour l'adapter à la tâche de détection de points clés de squelette multi-personnes dans des vidéos.

4.3.1 Architecture STM

STM (Space-Time Memory networks) est une architecture de réseaux de neurones profonds conçue pour la tâche de VOS (Video Object Segmentation). Elle fonctionne selon deux modes d'inférence : mémorisation et prédiction, tel qu'illustré dans la figure 4.7. La frame à segmenter est appelée "query frame". Les frames mémoires sont les frames passées avec leur masque de segmentation (issues de vérité terrain ou prédites par STM). Dans le mode mémorisation, le memory encoder prend comme entrée une frame RGB et le masque de segmentation d'objet (représenté par une carte de probabilité à un seul canal). Ils sont concaténés selon la dimension des canaux avant d'entrer dans le memory encoder, qui produit deux cartes de caractéristiques (key et value). Toutes les frames mémoires sont concaténées ensemble selon la dimension temporelle pour représenter la mémoire space-time. Dans le mode prédiction, le query encoder prend une frame RGB en entrée et produit deux cartes de caractéristiques (key k et value v). Les cartes de caractéristiques query ($k^Q \in \mathbb{R}^{H\times W\times C/8}$,$v^Q \in \mathbb{R}^{H\times W\times C/2}$ avec H la hauteur, W la largeur, C la dimension des caractéristiques de la carte de caractéristiques extraite du backbone) et memory ($k^M \in \mathbb{R}^{T\times H\times W\times C/8}$,$v^M \in \mathbb{R}^{T\times H\times W\times C/2}$ avec T le nombre de frames mémorisées) sont toutes deux obtenues à partir du quatrième étage d'un backbone ResNet-50 pré-entraîné sur ImageNet, qui est ensuite traité par deux couches de convolution parallèles qui réduisent la résolution des caractéristiques et servent de bottleneck. Pendant la phase de décodage, la paire key/value d'une frame query passe dans un module de mémoire space-time qui calcule des poids à partir des similarités entre la query key map et les memory key maps sur la dimension spatio-temporelle de la vidéo. Cette opération de lecture mémoire est illustrée dans la figure 4.8. Ensuite, une somme pondérée des valeurs mémoire est effectuée (ce qui correspond dans la figure à la deuxième multiplication de matrices). La valeur de la mémoire est récupérée grâce à une concaténation de la somme pondérée avec la query value. Le décodeur prend cette entrée et reconstruit le masque de segmentation de la frame query en appliquant plusieurs étapes de raffinage et de up-scaling. L'architecture STM originale est l'une des plus rapides pour la tâche de single-object VOS. Elle est également bien adaptée pour la tâche de VOS sur le dataset DAVIS-2017, dans lequel les vidéos contiennent tout au plus quelques objets à suivre. Cependant, l'architecture STM n'est pas conçue pour être mise à l'échelle facilement à de grands nombres d'objets, car chaque objet doit être traité indépendemment par une inférence de STM. Cela cause des temps de calcul et un usage mémoire plus important quand le nombre d'objets à suivre augmente. C'est pourquoi nous proposons dans la suite de ce manuscrit une modification de l'architecture STM pour suivre des key-points et arêtes de squelettes de personnes.

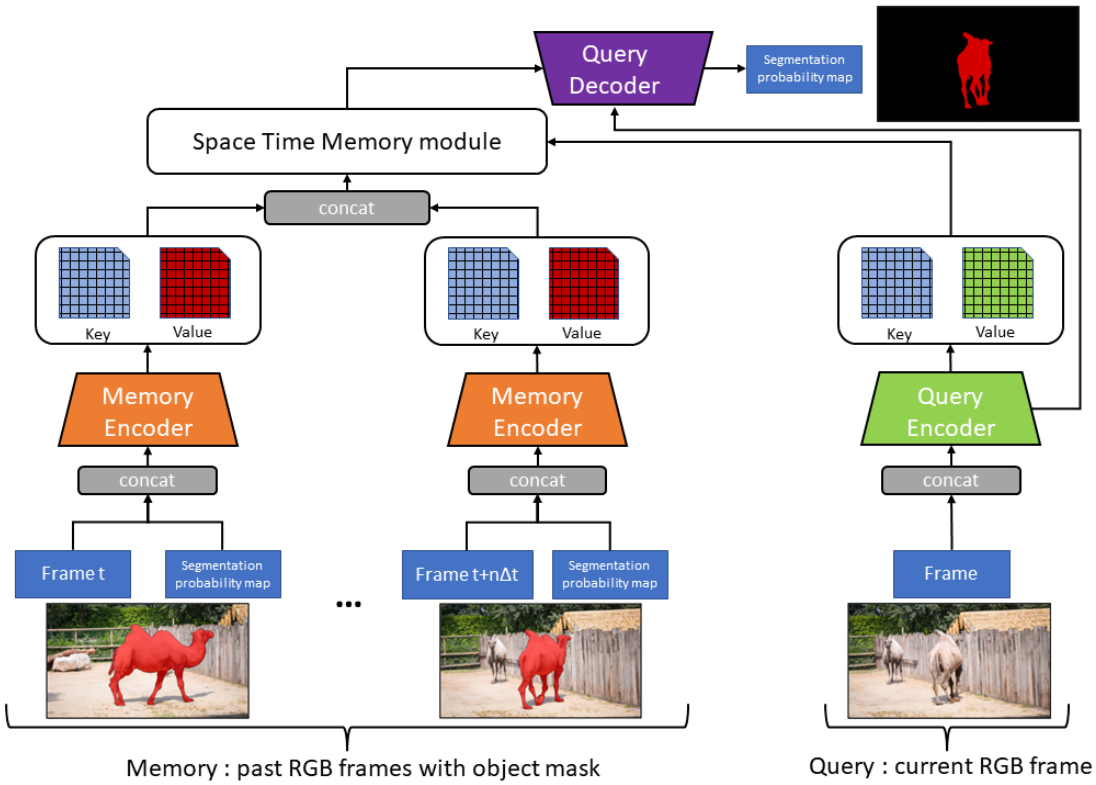


FIGURE 4.7 – Illustration de l'architecture STM [Oh *et al.*, 2019].

4.3.2 Architecture STMskeletons

Afin d'adapter l'architecture STM à la détection de points clés de squelette multi-personnes, nous avons modifié l'architecture STM afin que plusieurs squelettes puissent être traités en une seule inférence. Premièrement, pour être efficiente, l'architecture proposée doit être capable de produire plusieurs sorties, par contraste avec STM qui ne produit qu'une seule segmentation map. Deuxièmement, comme nous voulons détecter et suivre des parties de squelette, nous les représentons dans des canaux spécifiques à chacune. Cette nouvelle architecture est illustrée dans la figure 4.9. Les premières modifications concernent les entrées du memory encoder. L'architecture STM originale prenait en entrée une concaténation de la frame RGB et de la carte de segmentation correspondante. L'architecture proposée, nommée STMskeletons, représente les squelettes par leurs points clés et par les arêtes joignant deux points clés. Chaque point clé ou arête est représenté dans une carte de confiance dédiée : une par point clé de squelette et une par arête. Chaque carte de confiance couvre le point clé ou l'arête pour toutes les personnes présentes dans l'image, ce qui rend le temps de calcul et l'usage mémoire indépendant du nombre de personnes présentes dans la séquence vidéo. La valeur de chaque pixel comprise dans l'intervalle $[0; 1]$ donne la confiance sur la présence d'une des parties de squelette, parmi tous les squelettes présents dans l'image. 1 indique la certitude que le

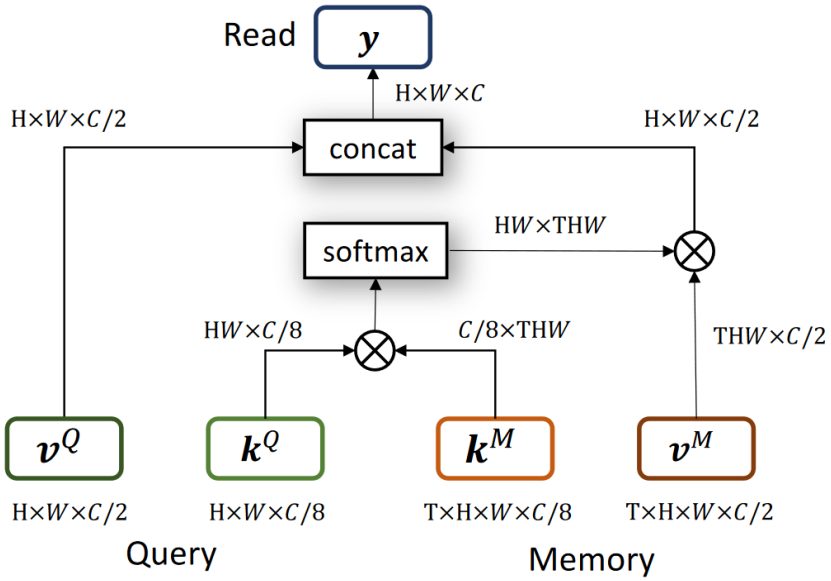


FIGURE 4.8 – Module de mémoire space-time. ⊗ représente une opération de multiplication de matrices.

point clé (ou l'arête) est présent à ce pixel, 0 indique qu'il ne l'est pas. Ainsi quatre entrées différentes sont concaténées :

- La frame RGB (comme dans STM) ;
- Une carte de probabilité d'objets représentée par l'union des cartes de points clés et d'arêtes de toutes les personnes présentes dans la frame (par contraste à STM qui ne considérait qu'une seule personne) ;
- N_K cartes de confiance qui représentent la probabilité de présence d'un point clé de squelette parmi toutes les personnes ;
- N_E cartes de confiance qui représentent la probabilité de présence d'une arête de squelette parmi toutes les personnes.

Étant donné que les cartes de confiance pour les arêtes ne contiennent pas d'information sur l'orientation de celle-ci comme dans les PAF [Cao *et al.*, 2019] il est plus approprié de les appeler des PAM (Part Affinity Maps), ce que nous ferons dans la suite de ce manuscrit.

Pour adapter le memory encoder à ces nouvelles entrées, nous avons modifié sa première couche. Ce changement dans le memory encoder est illustré dans la figure 4.10. Les entrées sont les canaux de la frame RGB, plus N canaux d'entrée ($N = N_K + N_E$ avec N_k étant le nombre de points clés du corps et N_E le nombre d'arêtes reliant deux points clés), plus une carte d'objet, créée à partir des canaux de points clés et d'arêtes

selon l'équation 4.10 suivante.

$$O = \prod_{i=0}^{N_K}(1 - K_i) \prod_{j=N_K+1}^{N_K+N_E}(1 - E_j) \tag{4.10}$$

avec K_i la carte de confiance sur la présence du point clé i et E_j la carte de confiance sur la présence de l'arête j. Ainsi la carte d'objet peut également s'appeler carte de background. Au final, les nouvelles dimensions d'entrées du memory encoder sont $(4 + N) \times H \times W$ ($N = N_K + N_E$, H étant la hauteur, W la largeur). Ainsi cette nouvelle première couche de convolution doit être entraînée à partir d'une initialisation aléatoire. res2, res3 et res4 font référence aux différents étages de l'architecture DNN ResNet-50 qui sont chacun constitués de plusieurs couches de convolution.

Le decoder doit également être adapté pour fournir la carte d'objets, et les cartes de confiance de points clés et d'arêtes. Le changement effectué dans le decoder est illustré dans la figure 4.11. La sortie du decoder original de STM était constituée de deux canaux, un donnant la probabilité de foreground et l'autre donnant la probabilité de background, ce qui implique des dimensions de $2 \times H/4 \times W/4$. Pour STMskeletons, il faut une probabilité de foreground et de background pour chacun des canaux de points clés ou d'arêtes, les nouvelles dimensions sont donc $N \times 2 \times H \times W$. La carte d'objets prédite par le modèle est calculée à partir des canaux de points clés et d'arêtes et n'implique donc pas de canal de sortie supplémentaire dans le decoder.

Les cartes de confiance fournies en entrées peuvent être des prédictions passées du modèle, ou bien être construites à partir d'annotations de vérité terrain ou à partir d'un détecteur single-frame de pose humaine.

4.3.3 Construction des cartes de confiance

Les cartes de confiance VT (Vérité Terrain) sont construites à partir des points clés et des arêtes VT. Pour une partie du corps j dans un squelette k à l'emplacement $x_{j,k}$, la valeur de la carte de confiance de point clé $S_{j,k}$ aux coordonnées du pixel p est donnée par l'équation 4.11 ci-dessous :

$$S_{j,k}(p) = exp\left(\frac{\|p - x_{j,k}\|}{\sigma}\right) \tag{4.11}$$

Les points clés de squelettes sont agrégés avec un opérateur max selon $S_j = max_k(S_{j,k})$. Pour les arêtes, nous construisons une image de base (qui sera recopiée pour chaque arête de corps humain) d'une arête I de largeur w et de largeur h avec une diffusion déterminée par σ. Pour un pixel de coordonnées (x, y) l'intensité du pixel est donnée par les relations 4.12 à 4.14 suivantes :

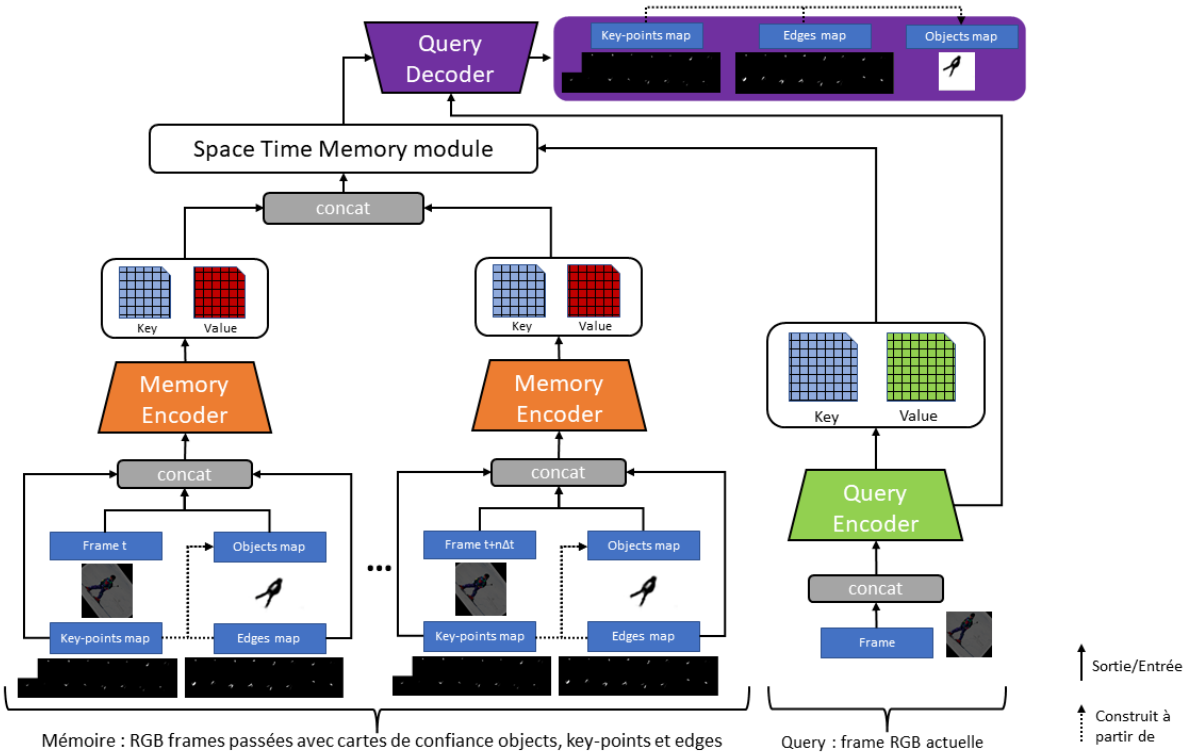


FIGURE 4.9 – Illustration de la nouvelle architecture STMskeletons proposée.

$$\text{Si } x \in [0.2w, 0.8w], \text{ alors } I(x,y) = exp\left(\frac{\|y - \frac{h}{2}\|}{\sigma}\right) \tag{4.12}$$

$$\text{Si } x < 0.2w, \text{ alors } I(x,y) = exp\left(\frac{\|(x,y) - (0.2w, 0.5h)\|}{\sigma}\right) \tag{4.13}$$

$$\text{Si } x > 0.8w, \text{ alors } I(x,y) = exp\left(\frac{\|(x,y) - (0.8w, 0.5h)\|}{\sigma}\right) \tag{4.14}$$

Cette image d'arête est illustrée dans la figure 4.12. Pour l'arête $E_{i,j,k}$ (l'arête entre les parties du corps i et j dans le squelette k) une rotation qui peut être suivie d'une mise à l'échelle est appliquée à l'image pour être positionnée entre les deux parties du corps i et j. Les arêtes sont ensuite aggrégées avec l'opérateur max similairement aux points clés.

Cette nouvelle architecture sacrifie légèrement la capacité de généralisation du modèle entraîné. En effet chaque canal de sortie est spécialisé pour un point clé ou arête de squelette particulier. Il ne peut donc pas être réutilisé pour suivre des objets autre que des point clés ou arêtes de personne, alors que c'était une possibilité offerte dans l'architecture originale. Cependant, ce changement réduit drastiquement l'usage mémoire

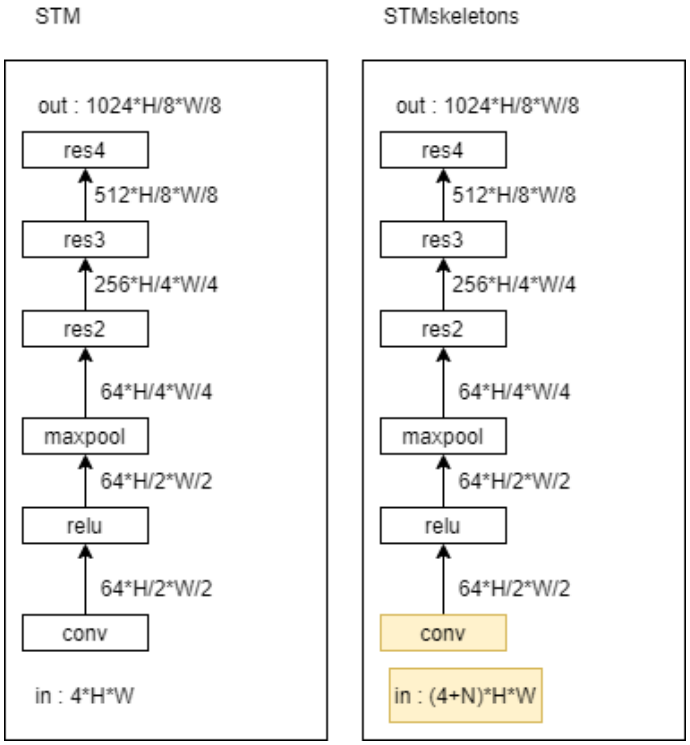


FIGURE 4.10 – Changements proposés dans le memory encoder de STM. Gauche : STM original. Droite : notre proposition de modifications en jaune donnant lieu au STMskeletons.

et le temps de calcul quand il s'agit de suivre 15 points clés de squelette, ce qui le rend indispensable pour un usage applicatif temps réel.

4.3.4 Entraînement

L'entraînement est réalisé avec un GPU Nvidia V100 et une taille de batch de 1. L'optimiseur utilisé est Adam, avec un taux d'apprentissage de 10^{-6}. Si ce n'est pas spécifié, la fonction de loss utilisée dans nos expériences est la Pearson Correlation, définie par l'équation 4.15 :

$$CC(Y, \hat{Y}) = \frac{\sigma(Y, \hat{Y})}{\sigma(Y)\sigma(\hat{Y})} \tag{4.15}$$

Cette loss est choisie par défaut car elle donne une précision satisfaisante comme expliqué dans la sous-section 4.4.2. Les poids officiels (obtenus avec un pré-entraînement sur plusieurs datasets d'images et affinés sur DAVIS 2017 comme décrit dans [Oh *et al.*, 2019]) sont utilisés pour l'initialisation de nos expérimentations sur les architectures STM et STMskeletons. Dans le cas de STMskeletons, seules les couches qui sont inchangées entre notre architecture et l'originale sont initialisées avec les poids officiels. De

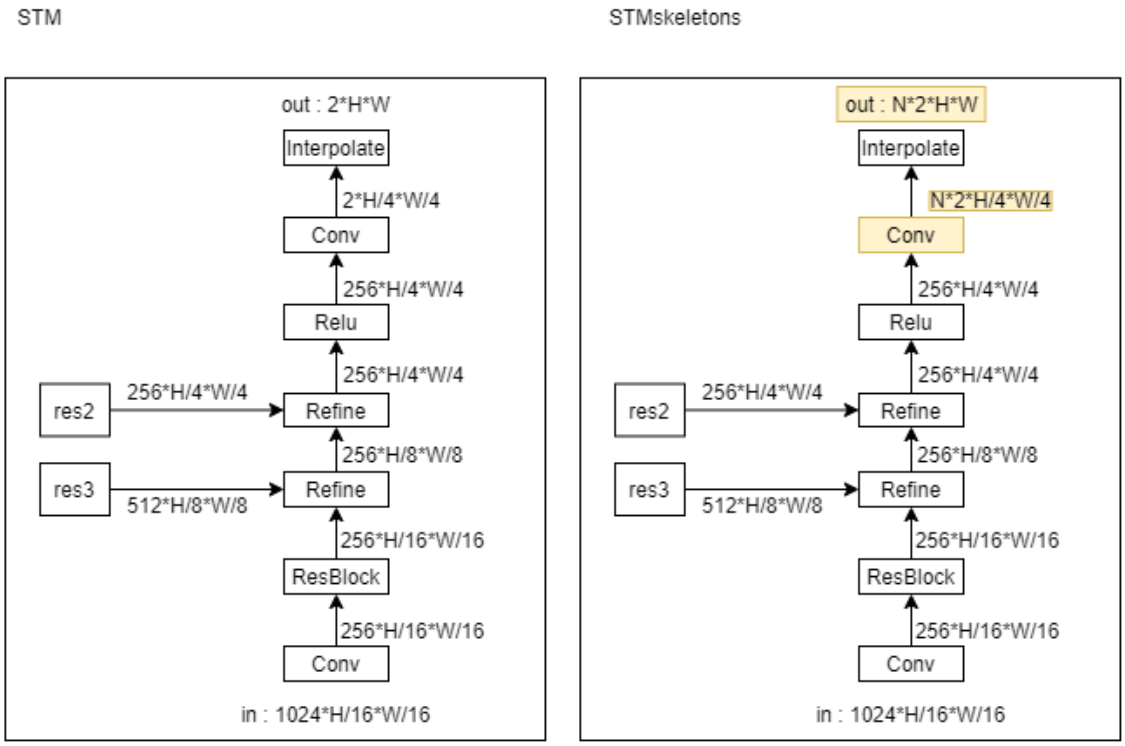


FIGURE 4.11 – Changements proposés dans le decoder de STM (en jaune). Gauche : STM original. Droite : notre proposition de modifications en jaune donnant lieu au STMskeletons.

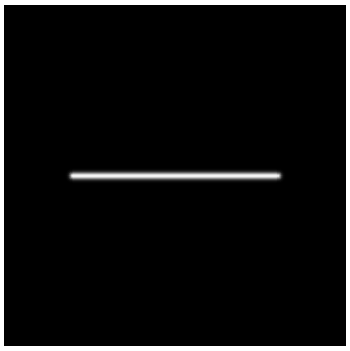


FIGURE 4.12 – Image d'arête.

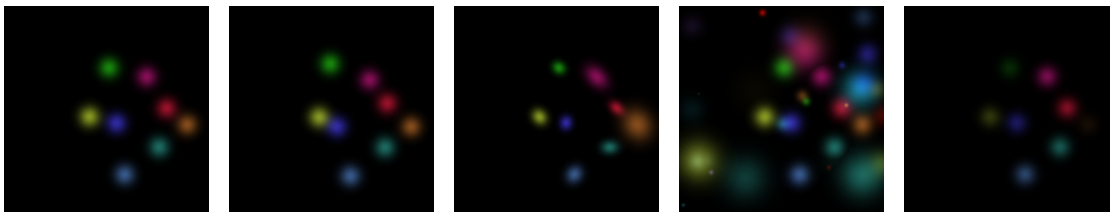
	Rouge	Vert	Bleu	Partie du corps
	166	11	0	nez
	6	1	97	oreille gauche
	24	135	14	oreille droite
	135	15	87	épaule gauche
	135	113	16	épaule droite
	153	18	43	coude gauche
	32	150	166	coude droit
	133	77	29	poignet gauche
	135	153	35	poignet droit
	48	42	163	hanche gauche
	130	51	35	hanche droite
	29	107	98	genoux gauche
	62	37	94	genoux droit
	56	89	138	cheville gauche
	138	61	148	cheville droite

FIGURE 4.13 – Couleur RGB de chaque point clé du corps d'une personne.

FIGURE 4.14 – Séquence synthétique créée à partir d'une seule image échantillonnée de MS-COCO keypoints.

plus, dans toutes nos expérimentations, les couches de batch normalization sont désactivées comme dans [Oh *et al.*, 2019].

Le dataset PoseTrack18 comporte une grande quantité de séquences vidéos, mais la diversité de personnes et de contextes est inférieure à des grands datasets d'images comme MS-COCO. Pour faire bon usage de la grande quantité d'images dans MS-COCO keypoints dans la tâche de détection de points clés de squelette dans des vidéos, nous avons créé des séquences vidéo synthétiques à partir d'images seules. Le code couleur utilisé pour les points clés du corps est illustré dans la figure 4.13. Pour créer une séquence de N frames, nous réalisons pour un échantillon donné une translation, une rotation et une mise à l'échelle, puis un cisaillement aléatoire de manière cumulative. Pour une meilleure visualisation, une séquence synthétique est illustrée dans la figure 4.14. Dans la suite de ce chapitre, les pré-entraînements sur MS-COCO_Train ont une durée de 5 époques.



(a) sans (b) jitter (c) rand (d) baits (e) dull_clouds

FIGURE 4.15 – De gauche à droite : illustration des points clés sans augmentation, puis avec jitter, rand, baits, et enfin dull_clouds

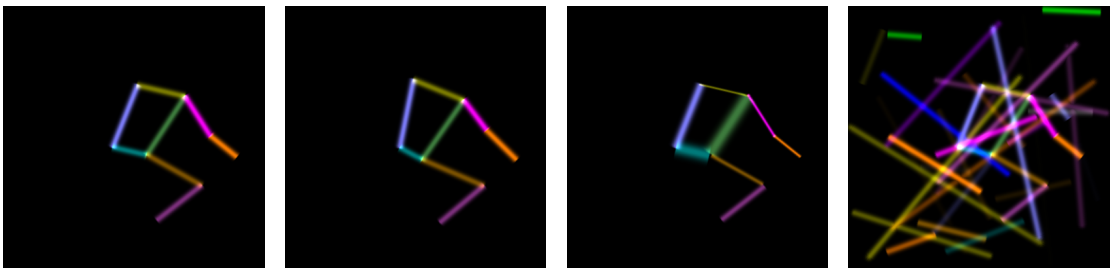
Nous avons ensuite affiné nos modèles sur le jeu de données PoseTrack18_Train qui contient de vraies séquences vidéos. Les échantillons d'entraînement sont créés en choisissant des vidéos au hasard, puis en sélectionnant un sous-ensemble de N frames dans la vidéo choisie, en gardant bien évidemment leur ordre chronologique. Différents programmes d'entraînement sont testés dans ce chapitre, pour 5, 30, ou 50 époques.

Pour le suivi à long terme, l'algorithme doit être capable de corriger les erreurs qu'il a commis, c.-à-d. les "raffiner", de telle manière que les prédictions de la frame T soient meilleures que les prédictions de la frame $T-1$. Cependant, les échantillons donnés au modèle pendant l'entraînement contiennent des annotations de vérité terrain (VT), qui ne comportent que peu d'erreurs. Pour préparer le modèle à gérer ces erreurs, nous avons conçu une méthode similaire à [Fieraru *et al.*, 2018]. Elle consiste à implémenter une stratégie d'augmentation de données pour le raffinement, où, pendant la construction des cartes de confiance des points clés et des arêtes, quelques transformations aléatoires listées ci-dessous sont appliquées.

- Application d'un léger déplacement aléatoire sur la position des points clés VT (appelé jitter) ;
- Randomisation de la taille, de la forme et de l'orientation des pics gaussiens des points clés, et randomisation de la taille des arêtes gaussiennes (appelé rand) ;
- Ajout de faux positifs à la fois aux points clés et aux cartes de confiance d'arêtes (appelés baits) pour que le modèle apprenne à les effacer de sa sortie ;
- Randomisation de l'intensité des pics gaussiens par multiplication d'un facteur aléatoire compris dans l'intervalle $[0, 1]$ (appelé dull_clouds).

Cette stratégie d'augmentation de donnée est illustrée dans les figures 4.15 et 4.16. Dans la section 4.5 nous examinons l'impact de ces options sur les performances de suivi.

Lorsqu'il est utilisé pour le suivi à long terme, le mode de mémorisation de notre modèle va prendre en entrée la prédiction produite pour les frames précédentes. Cependant, dans la procédure d'entraînement standard, les cartes de confiance qui sont



(a) sans (b) jitter (c) rand (d) baits

FIGURE 4.16 – De gauche à droite : illustration des arêtes sans augmentation, puis avec jitter, rand, et baits. L'augmentation dull_clouds n'est pas représentée car elle ne s'applique pas aux arêtes.

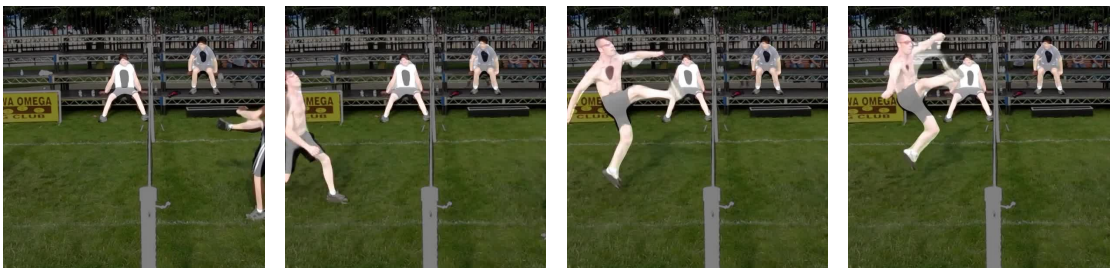
données au modèle sont créées à partir d'annotations de VT. Cette différence entre l'entraînement du modèle, et la manière selon laquelle il est sensé être utilisé pour le suivi à long terme, peut causer une dégradation des performances. Pour pallier ce problème, nous avons donc créé une procédure d'entraînement spécifique que nous appelons "cyclic". Dans cette procédure, pour une séquence échantillonnée, la mémoire est initialisée avec les annotations de la VT pour les quelques premières frames, puis, pour les dernières frames, utilise ses propres prédictions pour suivre les squelettes indépendamment de la VT. Dans cette procédure, le modèle peut produire des prédictions pour plusieurs frames, ainsi la loss d'entraînement est la somme de la loss de chaque prédiction.

4.4 Preuve de concept

Trois expérimentations (détaillées ci-après) ont alors été réalisées pour prouver que l'architecture STM, sans modification, peut être utilisée pour la tâche de détection de points clés de squelette dans des vidéos. Ces expériences permettent également de comparer différents paramètres d'entraînements et procédures, afin d'identifier ce qui est le plus adapté. Les métriques utilisées pour ces expériences sont : précision, rappel et F1-score. Nous ne pouvons pas, à ce stade, utiliser les métriques MOTA et mAP généralement utilisées pour un système de suivi complet car nous proposons à ce stade uniquement un élément d'un tel système.

4.4.1 Segmentation des squelettes

Le but de la première expérimentation est de prouver que l'architecture STM, conçue initialement pour segmenter des objets, peut être utilisée pour segmenter directement les squelettes de personnes. Ainsi, nous créons pour chaque frame dans l'échantillon d'entraînement, un masque de squelette, constitué d'une union de toutes les poses humaines annotées dans l'image. Nous essayons ensuite d'affiner un modèle STM, initialisé



(a) frame 1 (b) frame 42 (c) frame 75 (d) frame 100

FIGURE 4.17 – Résultats sur quelques frames d'un STM entraîné pour la segmentation binaire (squelette/fond).

à partir des poids officiels d'architecture STM, afin qu'il puisse suivre les squelettes de personnes dans des vidéos réelles. Pour ce test, nous exécutons un seul pré-entraînement sur le dataset MS-COCO keypoints, avec une fonction de loss cross-entropy. Nous évaluons le modèle entraîné sur le dataset PoseTrack18_Validation FULL, en initialisant le suivi avec la vérité terrain de la première frame, et en utilisant le modèle STM pour suivre le masque des squelettes sur le reste de la séquence vidéo. Nous calculons la précision et le rappel au niveau des pixels, en binarisant les prédictions du modèle avec un seuil de 0.5, et nous calculons le nombre de pixels considérés comme de vrais positifs (TP), faux positifs (FP) et faux négatifs (FN). Ensuite, nous calculons les métriques classiques précision, rappel et F1-score, à partir de TP, FP et FN obtenus précédemment ($précision = \frac{TP}{TP+FP}$, $rappel = \frac{TP}{TP+FN}$, $F1\text{-}score = 2 \cdot \frac{précision \cdot rappel}{précision+rappel}$). Pour cette première expérience, nous obtenons une précision de 58.9, un rappel de 85.8, et un F1-score de 71.7. Les résultats sont illustrés dans la figure 4.17. A partir de ces résultats visuels et qualitatifs, nous pouvons conclure que le modèle STM est capable de suivre correctement ce nouveau type de cible, malgré une conception adaptée au suivi d'objets génériques. Le modèle étant entraîné sur des données synthétiques avec le dataset MS-COCO keypoints et évalué sur des données vidéos réelles extraites du dataset PoseTrack18_Validation FULL, cette expérience montre aussi les capacités de généralisation de ce nouveau type d'algorithmes.

4.4.2 Cartes de confiance des squelettes

La deuxième expérience a deux objectifs. Le premier est de passer de l'usage d'un masque binaire à une carte de confiance dans l'intervalle $[0; 1]$ comme cible d'entraînement. Le deuxième est d'examiner l'impact de différentes fonctions de loss sur les performances du STM, car la cross-entropy n'est pas adaptée pour une cible non-binaire. Nous avons considéré le Pearson Correlation Coefficient (CC) et la Focal Loss (voir [Bruckert *et al.*, 2021] pour une vue d'ensemble de ces loss), cette dernière étant définie par

l'équation 4.16 ci-dessous :

$$FL(Y, \hat{Y}) = -\sum_{i=1}^{N}((1-\hat{Y}_i^{\gamma}) \times Y_i log(\hat{Y}_i) + \hat{Y}_i^{\gamma}(1-Y_i)log(1-\hat{Y}_i)) \tag{4.16}$$

La procédure d'entraînement est, à part ces précédents détails, la même que dans la première expérience. Les résultats sont illustrés dans la figure 4.18 et l'évaluation quantitative est illustrée dans le tableau 4.1. Dans ce tableau, nous montrons les résultats du sous-ensemble de validation de MS-COCO keypoints, sur des séquences short et full extraites du dataset PoseTrack18_Validation (ce qui constitue deux datasets nommés respectivement PoseTrack_Validation SHORT et PoseTrack_Validation FULL). Nous remarquons que la Focal loss obtient le meilleur F1-score sur MS-COCO_Validation (74.1 contre 71.8 pour la CC-Loss et 72.8 pour la MSE Loss) et PoseTrack18_Validation FULL (37.2 contre 34.3 pour la CC-Loss et 34.6 pour la MSE Loss) et se classe en deuxième positions sur PoseTrack18_Validation SHORT (50.6 contre 48.8 pour la MSE Loss et 50.9 pour la CC-Loss). La CC-loss obtient la meilleure précision sur MS-COCO_Validation (80.7 contre 79.1 pour la MSE Loss et 77.8 pour la Focal Loss) et sur PoseTrack18_Validation FULL (48.0 contre 46.8 pour la MSE Loss et 44.3 pour la Focal Loss). Ainsi il y a des avantages et inconvénients dans l'utilisation de chaque fonction de loss. Mais, les différences entre le plus haut score et le plus bas sont limitées. En effet, concernant le F1-score, elles ne sont respectivement que de 2.3, 2.1 et 2.9 sur MS-COCO_Validation, PoseTrack18_Validation SHORT et Posetrack18_Validation FULL. Pour la précision, elles ne sont que de 2.9, 1.5 et 3.7. Enfin pour le rappel, les valeurs sont de 6.2, 4.3 et 5.4. Les différences de performances apportées par les différentes Loss sont donc minimes. Les meilleures performances sont obtenues sur des données synthétiques du sous-ensemble de validation de MS-COCO keypoints, ce qui était attendu à cause de la similarité avec les données d'entraînement. A l'inverse, les moins bonnes performances sont obtenues pour le défi plus difficile PoseTrack18_Validation FULL. En conclusion, les résultats montrent que l'architecture STM est capable de traiter des cartes de confiance au lieu de masques de segmentation, et que le choix de fonction de loss n'a que peu d'importance.

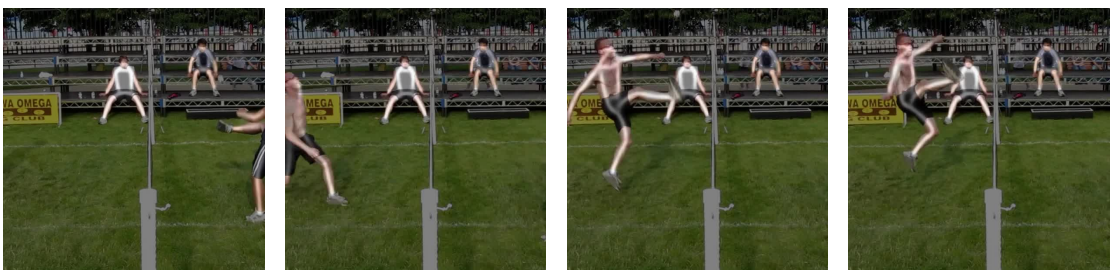
4.4.3 Détection d'arêtes de squelettes

Dans la troisième expérience, nous cherchons à prouver que l'architecture STM peut être utilisée pour différencier différents types d'arêtes qui connectent deux points clés. Ceci permet de montrer la pertinence d'affiner le modèle sur PoseTrack18_Train, et de comparer la CC-Loss avec la Focal Loss. Pour le premier but, le modèle doit traiter 16 cartes de confiance différentes, créées d'une manière similaire à la deuxième expérience, mais avec chaque type d'arête présent uniquement dans une carte de confiance particulière. Cela est nécessaire si STM vise à être utilisé pour la tâche de détection de

Expérience 2	Précision	Rappel	F1-score
	MS-COCO_Validation		
CC-Loss	**80.7**	64.6	71.8
MSE Loss	79.1	67.4	72.8
Focal Loss	77.8	**70.8**	**74.1**
	PoseTrack18_Validation SHORT		
CC-Loss	37.2	**80.3**	**50.9**
MSE Loss	36.5	73.8	48.8
Focal Loss	**38.0**	76.0	50.6
	PoseTrack18_Validation FULL		
CC-Loss	**48.0**	26.7	34.3
MSE Loss	46.8	27.5	34.6
Focal Loss	44.3	**32.1**	**37.2**

Tableau 4.1 – Comparaison des fonctions de loss dans l'entraînement du modèle STM pour prédire une carte de confiance de squelettes de personnes.

points clés de squelette dans des vidéos. Pour ce faire, nous avons lancé plusieurs pré-entraînements et affinages et évalué les modèles résultats sur MS-COCO keypoints, PoseTrack18_Validation SHORT, et PoseTrack18_Validation FULL. Les résultats sont illustrés dans la figure 4.19 et dans le tableau 4.2. Chaque canal de carte de confiance d'arête a une couleur assignée. Nous choisissons de ne comparer que la CC-loss et la Focal Loss car l'expérience précédente a montré que la CC-loss obtient la meilleure précision et la Focal Loss le meilleur rappel. Sur MS-COCO_Validation, nous remarquons que le meilleur F1-score est obtenu avec un modèle pré-entraîné avec la Focal Loss (80.7 contre 34.6, 45.7, 51.4 et 75.0). Il en est de même sur PoseTrack18_Validation FULL, avec un modèle affiné sur 30 epochs avec la Focal Loss (30.1 contre 23.8, 27.5, 27.7 et 22.1). Sur Pose-



(a) frame 1 (b) frame 42 (c) frame 75 (d) frame 100

FIGURE 4.18 – Résultats sur quelques frames d'un STM entraîné à produire des cartes de confiance pour la présence de squelettes de personnes.

Track18_Validation SHORT, il s'agit d'un modèle pré-entraîné avec la CC-Loss (47.7 contre 45.5, 45.2, 42.6, et 47.1). Nous constatons donc que l'affinage sur PoseTrack18_Train semble améliorer les performances avec la CC-loss sur MS-COCO_Validation (F1-score de 51.4 avec 30 epochs d'affinage contre 45.7 avec 5 epochs d'affinage et 34.6 sans affinage) tout comme sur PoseTrack18_Validation FULL (F1-score de 27.7 avec 30 epochs contre 27.5 avec 5 epochs et contre 23.8 sans affinage). Le F1-score semble également être amélioré par l'affinage avec la Focal Loss sur PoseTrack18_Validation SHORT (47.1 contre 42.6) et sur PoseTrack18_Validation FULL (30.1 contre 22.1). Cependant l'affinage semble diminuer le F1-score avec la Focal Loss sur MS-COCO_Validation (75.0 avec affinage contre 80.7 sans affinage). Etant donné que les performances sur PoseTrack18_Validation FULL sont les plus pertinentes pour un usage réel, nous considérons que ces résultats montrent l'intérêt de l'affinage sur PoseTrack18_Train. Cependant, il semble que les gains soient décroissants pour des affinages plus longs, car l'entraînement sur 30 époques n'améliore que très peu (amélioration <1%) les performances sur les différentes métriques.

Expérience 3	**Precision**	**Recall**	**F1-score**
	MS-COCO_Validation		
CC-Loss pré-entraîné	77.8	22.2	34.6
CC-Loss affiné 5 epochs	95.2	30.1	45.7
CC-Loss affiné 30 epochs	**95.4**	35.2	51.4
Focal Loss pré-entraîné	88.1	**74.3**	**80.7**
Focal Loss affiné 30 epochs	88.5	65.1	75.0
	PoseTrack18_Validation SHORT		
CC-Loss pré-entraîné	42.6	54.1	**47.7**
CC-Loss affiné 5 epochs	36.1	61.5	45.5
CC-Loss affiné 30 epochs	36.4	59.8	45.2
Focal Loss pré-entraîné	**53.0**	35.6	42.6
Focal Loss affiné 30 epochs	37.5	**63.1**	47.1
	PoseTrack18_Validation FULL		
CC-Loss pré-entraîné	33.5	18.5	23.8
CC-Loss affiné 5 epochs	25.1	30.4	27.5
CC-Loss affiné 30 epochs	25.4	30.5	27.7
Focal Loss pré-entraîné	**35.2**	16.1	22.1
Focal Loss affiné 30 epochs	29.0	**31.2**	**30.1**

Tableau 4.2 – Comparaison de différents modèles sur la tâche de production de cartes de confiance pour chacune des 16 arêtes du squelette humain.

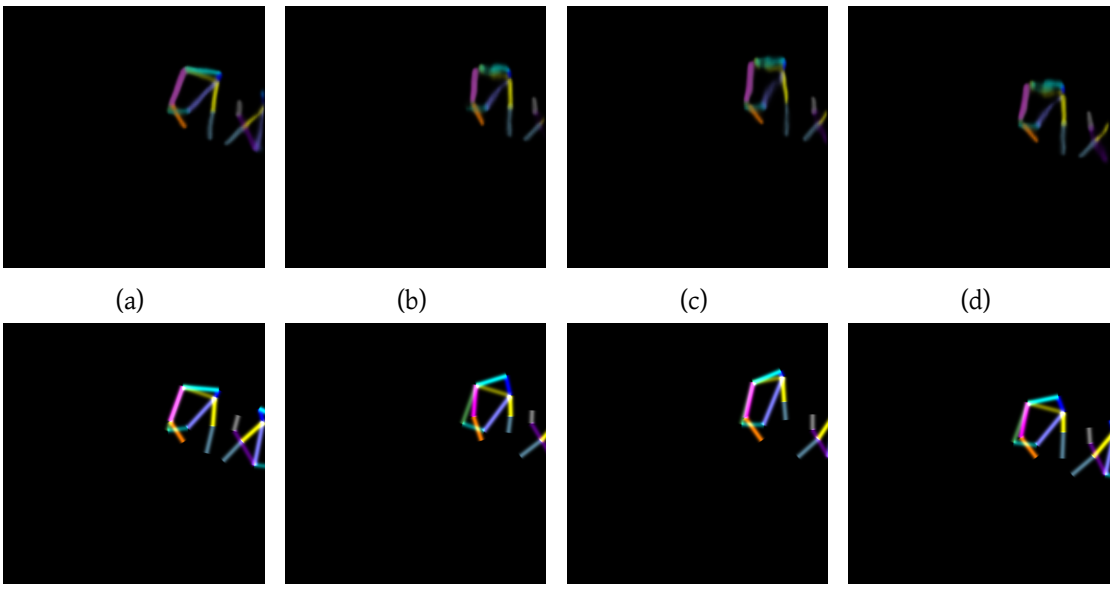


FIGURE 4.19 – Résultats de STM entraîné pour produire une carte de confiance pour chaque type d'arête de pose humaine. Première ligne : prédiction. Deuxième ligne : vérité terrain.

4.5 STMskeletons : une étude ablative

Les expérimentations que nous présentons maintenant ont été réalisées avec la nouvelle architecture STMskeletons, dédiée au suivi multi-personnes. Nous comparons dans cette section différentes procédures d'entraînement et d'augmentation de données pour STMskeletons.

Nous analysons l'impact de différents choix de configurations d'entraînement, et options pour l'augmentation de données comme présentées dans la section 4.3.4. Les modèles sont pré-entraînés pour 5 époques sur MS-COCO keypoints (32000 steps), et finetunés pour 50 époques sur PoseTrack18_Train (27500 steps).

L'évaluation est effectuée sur PoseTrack18_Validation FULL et nous utilisons, pour ce faire, une métrique spécifique. Nous associons chaque point clé avec la prédiction qui lui correspond le mieux (la plus proche), et nous considérons qu'il s'agit d'un vrai positif si la prédiction se situe dans un rayon de 10 pixels de la vérité terrain. Les arêtes ne sont pas considérées dans les résultats d'évaluation.

Les résultats sont illustrés dans le tableau 4.3. Nous comparons 7 configurations, avec ou sans affinage sur PoseTrack18_Train SHORT. La configuration 1 n'a aucune option activée. La configuration 2 a les options rand, baits et jitter activées. La configuration 3 a l'option cyclic activée. La configuration 4 a les options cyclic et rand activées. La configuration 5 a les options cyclic, rand, baits et jitter activées. La configuration 6 a toutes les options activées. La configuration 7 a les options cyclic, baits, et jitter activées. Quand

nous comparons la même configuration, avant et après l'affinage, nous remarquons une amélioration du F1-score. En effet, la configuration 1 passe de 11.4 à 29.4. La configuration 2 passe de 35.7 à 44.4. La configuration 3 passe de 37.2 à 46.8. La configuration 4 passe de 35.2 à 46.0. La configuration 5 passe de 38.9 à 47.6. La configuration 6 passe de 35.0 à 44.0. Enfin la configuration 7 passe de 38.9 à 47.9. Ces résultats montrent l'importance de l'affinage sur des séquences vidéos réelles de PoseTrack18_Train. En comparant les configurations 1 et 3, nous pouvons observer un gain significatif apporté par l'entraînement cyclique (le F1-score passe de 11.4 à 37.2 sans affinage, et de 29.4 à 46.8 avec affinage). Cette observation se confirme en comparant les configuration 2 et 5 (le F1-score varie de 35.7 à 38.9 sans affinage, et de 44.4 à 47.6 avec affinage). L'entraînement cyclique apporte donc une amélioration significative, et suffit presque à remplacer l'augmentation de données. L'option rand prise isolément semble dégrader les résultats (le F1-score diminue de 37.2 à 35.2 sans affinage, et de 46.8 à 46.0 avec affinage). Comparé à la configuration 5, la configuration 7 obtient de meilleurs résultats avec affinage en désactivant l'option rand (F1-score de 47.9 contre 47.6). L'option dull_clouds semble diminuer les résultats (F1-score diminué de 38.9 à 35.0 sans affinage et de 47.9 à 44.0 avec affinage). Cependant, baits, et jitter semblent être des options utiles qui permettent d'obtenir une amélioration consistante, en particulier quand l'entraînement cyclique est désactivé. Ces options sont nécessaires pour obtenir la meilleure performance mesurée en terme de F1-score, obtenue après l'affinage avec la configuration 7. Les résultats obtenus avec les meilleures configuration, 3, 4, 5, 6 et 7 sont illustrés sur 4 séquences d'images dans les figures 4.20 à 4.23. Les images présentées correspondent aux frames 5, 15, 25 et 35. Le suivi est initialisé à la première frame par les poses de la vérité terrain. Dans la figure 4.23, nous pouvons observer la capacité du modèle à détecter une personne qui n'était pas présente à la frame initiale et à raffiner progressivement sa prédiction grâce à son entraînément adapté.

En conclusion, les options d'augmentation de données rand et dull_clouds sont considérées comme néfastes pour les performances du modèle, tandis que les options cyclic, baits et jitter amènent de réelles améliorations. La configuration 7 est donc retenue comme étant la meilleure grâce au F1-score qu'elle obtient.

Étude ablative	cyclic	rand	baits	jitter	dull_clouds	Précision	Rappel	F1-score
SANS affinage sur PoseTrack18_Train SHORT						**PoseTrack18_Validation FULL**		
Configuration 1						**82,6**	6.1	11.4
Configuration 2		X	X	X		57.4	25.9	35.7
Configuration 3	X					67.3	25.7	37.2
Configuration 4	X	X				46.9	28.2	35.2
Configuration 5	X	X	X	X		52.1	**31.0**	**38.9**
Configuration 6	X	X	X	X	X	48.7	27.3	35.0
Configuration 7	X		X	X		62.8	28.1	**38.9**
AVEC affinage sur PoseTrack18_Train SHORT						**PoseTrack18_Validation FULL**		
Configuration 1						**81.3**	17.9	29.4
Configuration 2		X	X	X		73.1	31.8	44.4
Configuration 3	X					70.9	35.0	46.8
Configuration 4	X	X				62.3	**36.4**	46.0
Configuration 5	X	X	X	X		69.8	36.2	47.6
Configuration 6	X	X	X	X	X	69.8	32.2	44.0
Configuration 7	X		X	X		73.5	35.5	**47.9**

Tableau 4.3 – Comparaison de différentes configurations de pré-entraînement, avec différentes options d'augmentation de données pour le raffinement.

La précision, le rappel et le F1-score sont des métriques intéressantes pour comparer les performances de différentes procédures d'entraînement. Cependant, elles n'indiquent pas si les différences de performances se font sur le suivi à court ou à long terme. Pour être certain que notre procédure d'entraînement soit avantageuse pour le suivi à long terme, nous comparons dans la figure 4.24, l'évolution des scores de rappel en fonction du temps, sur toutes les séquences de PoseTrack18_Validation FULL. Nous constatons que les différences sur les premières frames sont inférieures à 5%, et donc négligeables, et qu'elles augmentent avec le temps. Ceci montre que l'affinage sur PoseTrack18_Train et l'entraînement cyclique améliorent les performances sur le long terme plutôt que sur le court terme.

FIGURE 4.20 – Comparaison qualitative des différentes configurations sur quelques frames extraites d'une vidéo de PoseTrack18_Validation FULL (ligne 1 : vérité terrain ; lignes 2-6 : configurations 3, 4, 5, 6 et 7).

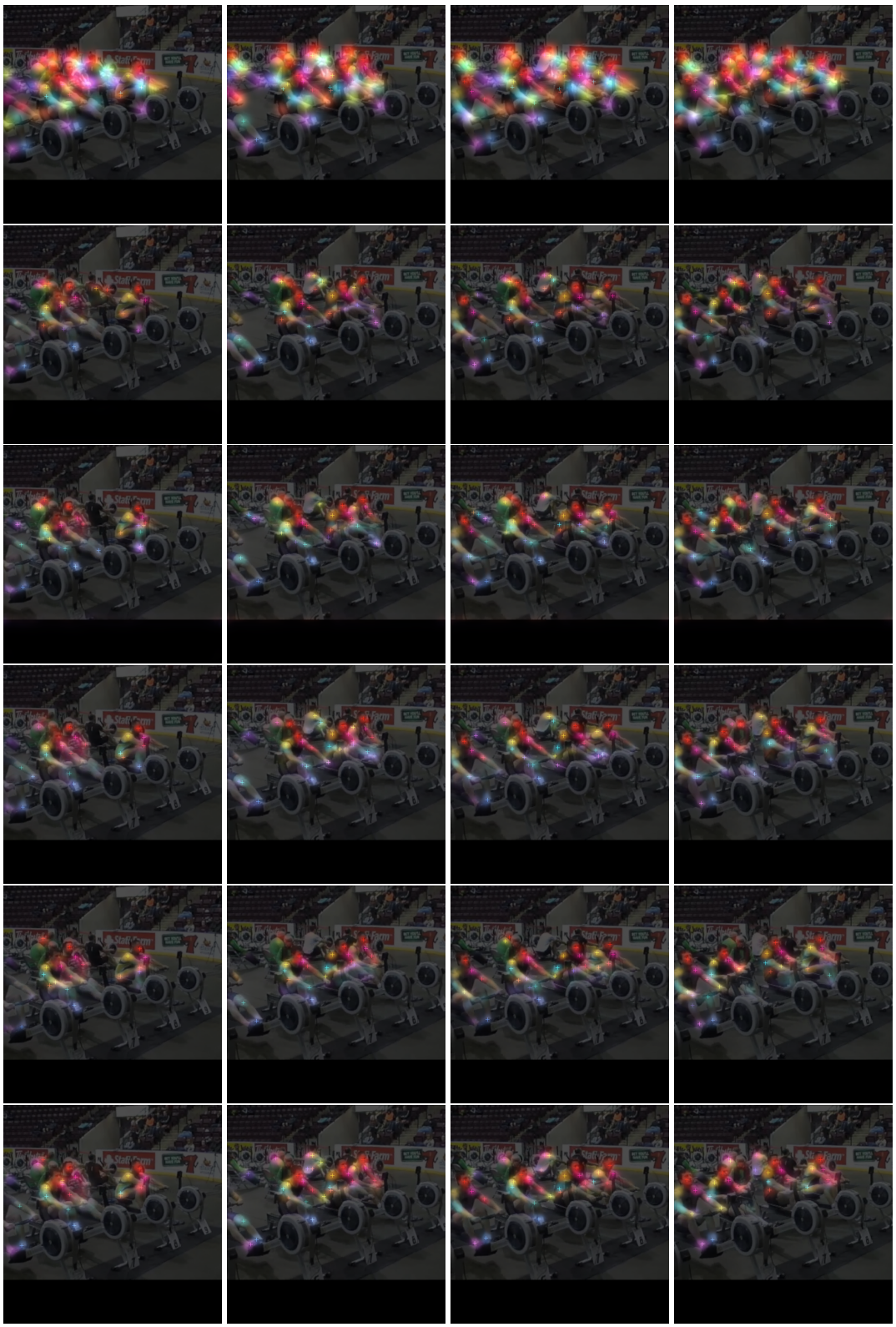


FIGURE 4.21 – Comparaison qualitative des différentes configurations sur quelques frames extraites d'une vidéo de PoseTrack18_Validation FULL (ligne 1 : vérité terrain ; lignes 2-6 : configurations 3, 4, 5, 6 et 7).

FIGURE 4.22 – Comparaison qualitative des différentes configurations sur quelques frames extraites d'une vidéo de PoseTrack18_Validation FULL (ligne 1 : vérité terrain ; lignes 2-6 : configurations 3, 4, 5, 6 et 7).

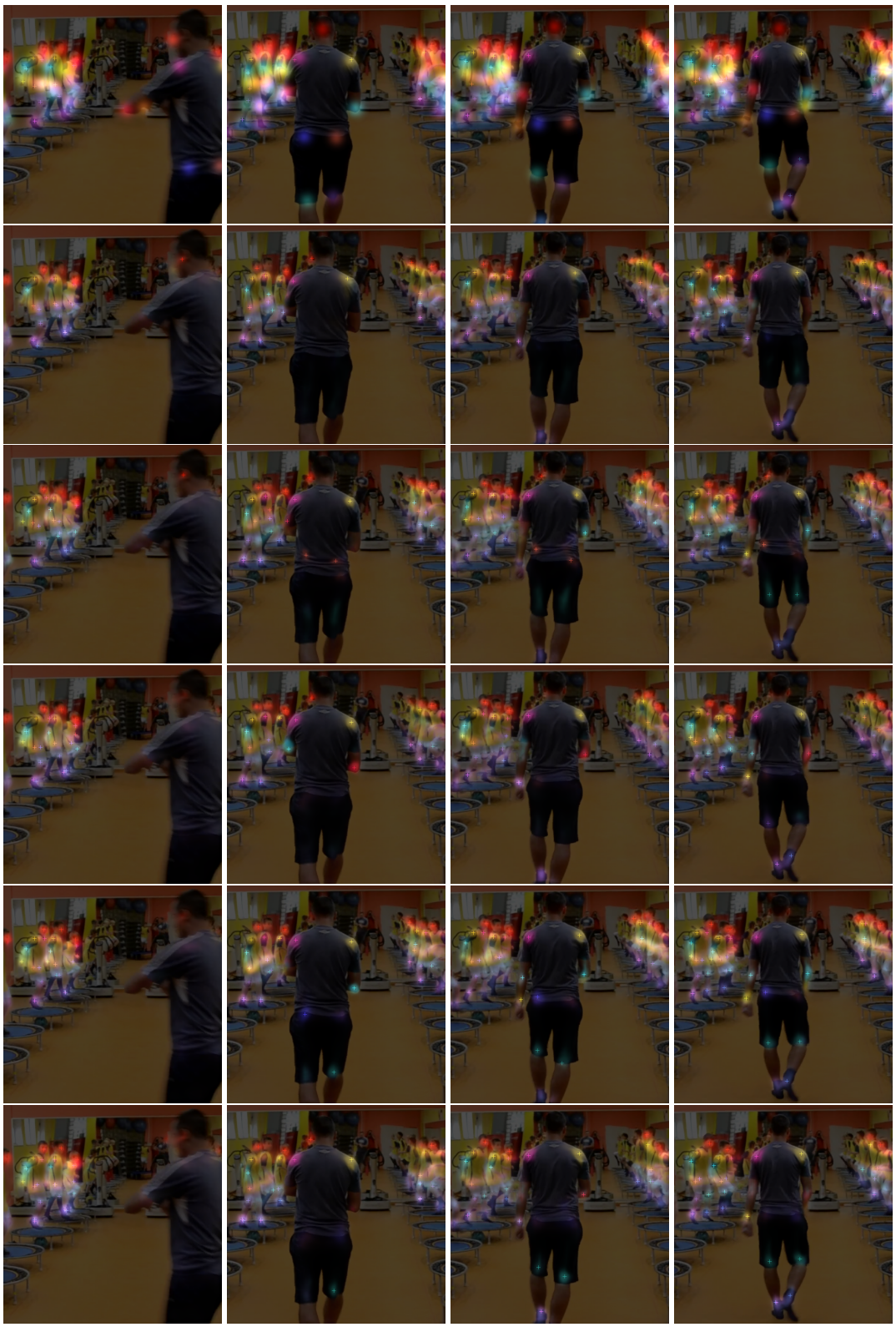


FIGURE 4.23 – Comparaison qualitative des différentes configurations sur quelques frames extraites d'une vidéo de PoseTrack18_Validation FULL (ligne 1 : vérité terrain ; lignes 2-6 : configurations 3, 4, 5, 6 et 7).

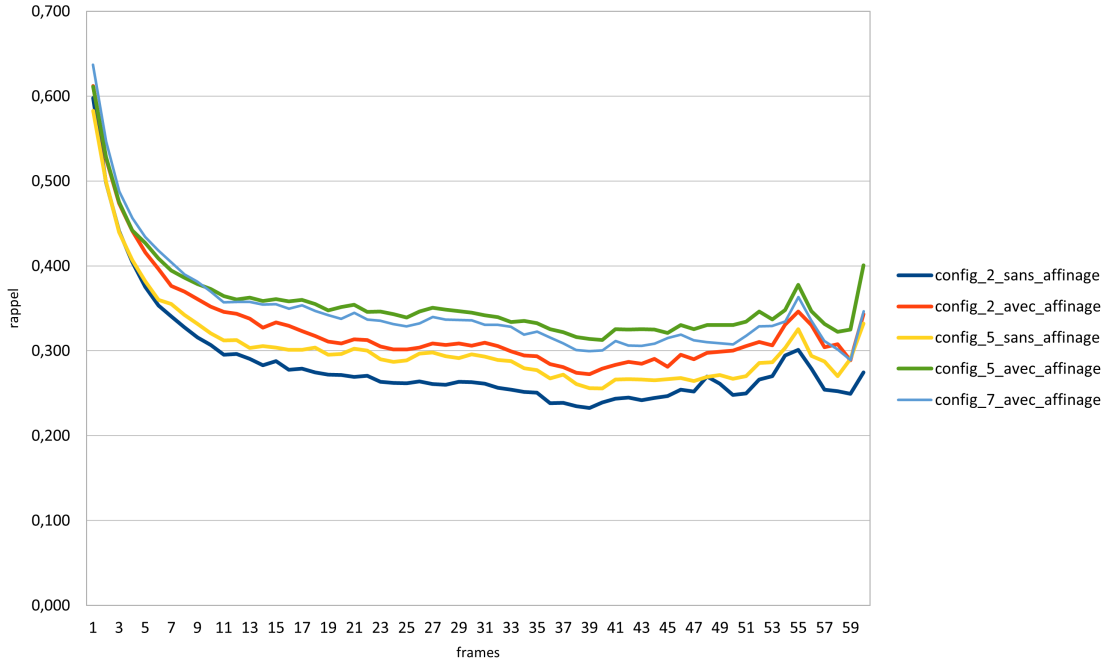


FIGURE 4.24 – Évolution du rappel dans le temps en fonction des différentes configurations.

4.6 Conclusion

Dans ce chapitre, nous avons décrit une méthode de détection de parties du corps (points clés) multi-personnes dans des vidéos. Cette méthode est inspirée de l'estimateur de pose multi-personnes OpenPose, et de l'algorithme de Video Object Segmentation STM. L'architecture initiale de STM a donc été modifiée afin de pouvoir traiter toutes les parties du corps de toutes les personnes dans l'image et ce, en une seule inférence. Tout d'abord, diverses expériences prouvant la faisabilité de notre méthode ont été réalisées. Ensuite, une étude ablative a été réalisée afin de comparer les augmentations de données et les méthodes d'entraînement et ainsi définir celles qui sont utiles pour les performances en suivi à long terme des différentes parties du corps d'une personne. Un entraînement cyclique avec des perturbations aléatoires de la position des parties du corps, et la présence de pièges (baits), a été déterminée comme la configuration la plus avantageuse (avec un F1-score de 47.9 sur PoseTrack18_Validation FULL, comparé avec un F1-score de 29.4 obtenu avec une configuration sans option). Nous montrons que la différence de performances entre différentes options n'est pas dans le suivi à court terme mais bien dans le suivi à long terme des parties du corps, et correspond donc au but recherché.

Le travail proposé dans ce chapitre a fait l'objet d'une publication [Dufour *et al.*, 2022].

Chapitre 5

Applications de vidéoprotection pour le futur train autonome

5.1 Contexte

Face aux avancées récentes dans les systèmes de transport intelligents, les premiers trains autonomes ont été déployés pour le transport de fret. En Australie, un train autonome de type GoA4 relie depuis le 10 juillet 2018 la mine de fer de Tom Price et le port de Cap Lambert pour transporter 28000 tonnes de minerais à 80 km/h sur une distance de 280 kms [Cognasse, 2018].

En France, la SNCF a lancé un programme de recherche sur le train autonome en 2016. Ce type de train permettrait une ponctualité améliorée, un trafic plus fluide, une consommation d'énergie réduite et plus de circulation. Deux volets composent ce programme : le volet Train de Fret Autonome, et le volet Train Autonome Service Voyageurs (TASV). Le volet TASV, concerné par cette thèse, a pour ambition de produire un prototype de train autonome passager de type GoA4 à l'horizon 2023 pour une industrialisation à partir de 2025.

Ce chapitre vise donc à appliquer les méthodes développées dans les deux chapitres précédents au contexte particulier du train autonome. Nous présentons les analyses quantitatives et qualitatives sur les performances de ces méthodes testées sur des vidéos annotées et acquises dans un train de type Regio2N.

5.1.1 Le programme Train Autonome

Les systèmes de transports autonomes sont devenus récemment un des centres d'intérêts majeur dans le domaine de l'intelligence artificielle. En effet, ils ont le potentiel d'apporter des avantages majeurs en termes de sécurité et de diminution des congestions. Parmi les systèmes de transports autonomes existants, les voitures autonomes ont reçu le plus d'attention ces dernières années et sont devenues le domaine de re-

cherche le plus populaire. Par voie de conséquence, et grâce aux avancées des technologies du Deep learning, elles atteignent maintenant le niveau 3 d'autonomie (conduite autonome limitée selon la classification SAE J3016). Récemment, les trains autonomes ont également intéressé plusieurs pays et entreprises. Selon l'Union Internationale des Transports Publics (UITP), ils peuvent être classés en 5 niveaux d'autonomie que nous présentons ci-dessous :

- GoA 0 : aucune autonomie (niveau SAE 0) ;
- GoA 1 : train à conduite manuelle contrôlée dotée d'un contrôle de vitesse par balise (KVB en France) interdisant les survitesses. (niveau SAE 1) ;
- GoA 2 : train à conduite semi-automatique. L'accélération et le freinage sont automatisés. Le conducteur gère les portes, le départ. Il reste responsable de la sécurité et peut reprendre le contrôle si besoin. (niveau SAE 2) ;
- GoA 3 : train totalement automatisé, capable de détecter les obstacles et les problèmes survenants dans l'environnement. Cependant, un agent reste présent dans le train pour intervenir auprès des passagers et en cas de situation exceptionnelle. (niveaux SAE 3 et 4) ;
- GoA 4 : totalement automatisé et sans personnel à bord, et donc capable de gérer toutes les situations. (niveau SAE 5).

Le programme Train Autonome consiste à produire un prototype de train atteignant GoA4 à l'horizon 2023. La figure 5.1 illustre le train utilisé. Ce projet se divise en 28 lots. La figure 5.3 illustre l'articulation entre ces derniers.

Le lot au sein duquel s'est déroulé cette thèse est le lot numéro 15 intitulé "Relation avec les voyageurs", qui a pour ambition de prendre en charge les relations avec les voyageurs dans un train sans personnel naviguant à bord. Ce lot se divise en un volet "service" pour lequel les voyageurs peuvent interagir avec le train, et un volet "surveillance" pour lequel le train assure la sécurité des voyageurs, et qui constitue le travail réalisé dans cette thèse.

Le modèle de train qui servira de prototype pour le futur train autonome est une version modifiée du Regio2N illustré dans la figure 5.2. C'est sur ce type de véhicule que nous travaillons et sur lequel les contributions des chapitres 3 et 4 seront évaluées pour être présentées dans ce dernier chapitre du mémoire.

5.1.2 Surveillance par apprentissage automatique

Dans le futur train autonome, chaque action réalisée par un personnel à bord ou à quai devra être à terme soit modifiée, soit réalisée par l'IA. A ce stade du projet, il

FIGURE 5.1 – Futur train autonome.

FIGURE 5.2 – Train de type Regio2N.

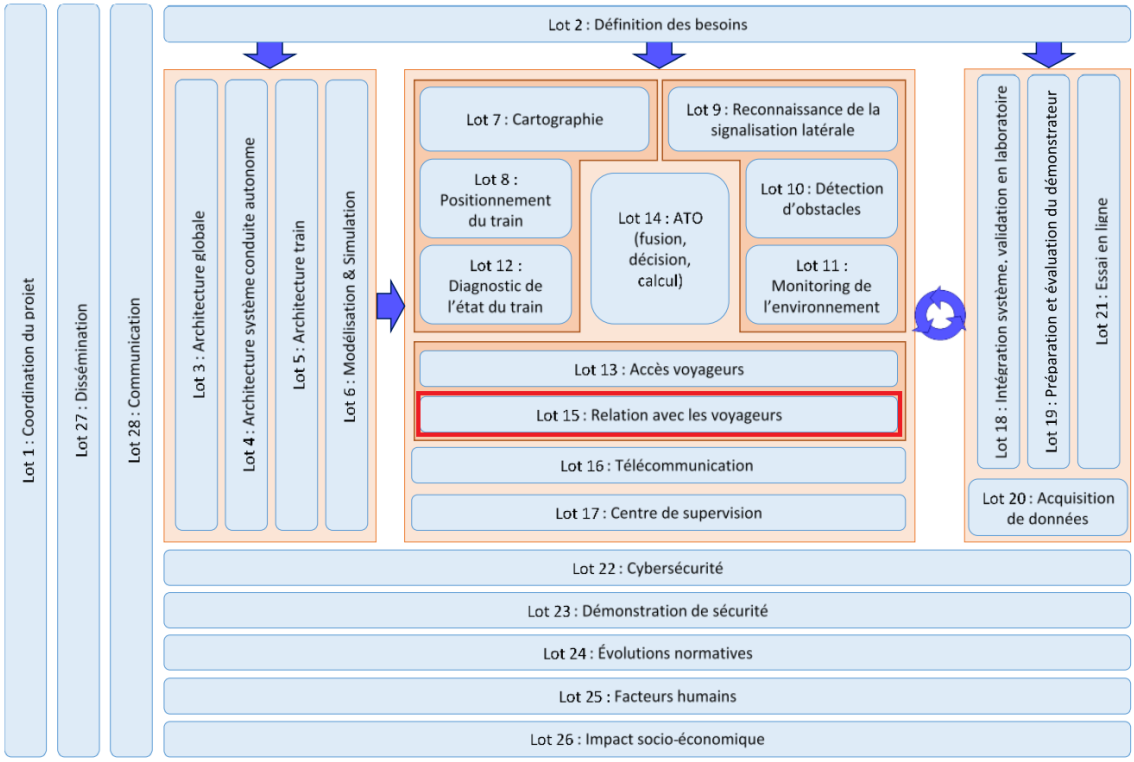


FIGURE 5.3 – Articulation des différents lots du programme Train Autonome Service Voyageurs piloté par l'IRT Railenium.

a été décidé que tout sera mis en œuvre pour éviter une modification des procédures existantes.

Les tâches du personnel présent à bord peuvent être classées en quatre catégories :

- Perception : le personnel voit et entend les passagers et ce qui se passe à l'intérieur des voitures ;
- Analyse : le personnel est capable de tirer des conclusions à partir de sa perception de la scène (voiture et passagers) ;
- Communication : le personnel est capable de communiquer avec les passagers, par exemple, pour répondre à leurs questions ;
- Action : le personnel a le pouvoir d'agir dans des situations qui le nécessitent (routinières ou exceptionnelles).

Dans ce manuscrit, nous allons nous concentrer sur la brique perception, indispensable pour que l'IA soit capable de prendre des bonnes décisions. Nous allons aussi détailler les besoins de haut-niveau qui seront nourris par cette brique en perception.

Perception

Plusieurs moyens peuvent être mis en oeuvre pour que le train puisse percevoir son environnement, analyser les situations rencontrées, et agir en conséquence, notamment en informant les usagers :

- Capteurs : caméras, lidars, microphones, capteurs de présence ;
- Communication : verbale (voix ou texte) ou non verbale (gestes) ;
- Système d'information : données récoltées par des systèmes externes (billeterie, Systèmes de positionnement satellitaires, centrale d'information voyageur, ...).

Dans le cadre de ce projet, seulement certaines situations de vie devront être détectées automatiquement. Une application mobile et éventuellement une application sur borne interactive sont aussi prévues pour permettre aux différents acteurs (agents, passagers, ...) de signaler des situations particulières ou évènements potentiellement dangereux.

Pour la détection des situations de vie, le système devra à minima être capable de percevoir les cas suivants :

- Position des passagers ;
- Posture (allongée, assise, debout) ;
- Identification unique (suivi) ;
- Présence de bagage ;
- Présence d'objet encombrant (vélo, poussette) ;
- Personnes à mobilité réduite (PMR).

Dans ce manuscrit nous nous concentrons sur la détection de la position et de la posture des passagers.

Analyse

Une fois la tâche de perception réalisée, les agents doivent identifier la situation et réagir de manière appropriée. A partir des données brutes de perception, il faut que l'IA se représente l'état du monde en consolidant les informations suivantes :

- Identification des groupes ;
- Association bagages / personnes ;

- Identification de situations spécifiques (bagarre, évanouissement, chute, blocage des portes).

Une fois le type de situation déterminé, l'IA prend une décision en priorisant les actions selon l'urgence et le niveau sécuritaire et en adaptant son comportement aux paramètres de la situation.

5.1.3 Motivation

Dans le cadre de notre application de surveillance à bord du futur train autonome, la première étape nécessaire à l'analyse de la scène captée par les caméras consiste à passer de la valeur RGB des pixels à une représentation qui contient des informations pertinentes sur la scène, en particulier sur la position et la posture des personnes présentes dans le train. Dans la littérature, les défis associés à ce type de tâche sont la détection et la segmentation de personnes, ainsi que l'estimation de pose. Ce sont les deux défis que nous traitons dans la suite de ce chapitre au travers :

- Le développement d'une méthode de détection de personnes efficiente sur des images grand angle (rectilinéaires) ou Fisheye (présentant des distorsions en barillet);
- Le développement d'une méthode de détection de points clés de squelette dans des vidéo avec une mémoire à long terme.

5.2 Perception vidéo à bord du futur Train Autonome

Précédemment, nous avons basé nos expérimentations sur différents datasets publics : ImageNet, MS-COCO et PoseTrack. Ces datasets, bien que très intéressants, ne contiennent cependant que peu d'images présentant les caractéristiques que nous retrouverons dans les images acquises à bord du futur train autonome. Pour pallier à ce problème, nous avons enregistré notre propre dataset dans un train Regio2N à l'arrêt au technicentre SNCF de Lille. Nous avons établi une liste de scénarios nécessaires pour évaluer les algorithmes dans une grande variété de situations réelles, que nous détaillerons en sous-section 5.2.2.

5.2.1 Dispositif vidéo

Les scénarios présentés ont été enregistrés par des caméras installées dans le train Regio2N aux emplacements indiqués dans la figure 5.4. Nous pouvons y voir, une caméra à chaque porte de la voiture, et deux autres placées à chaque extrémités de la rangée

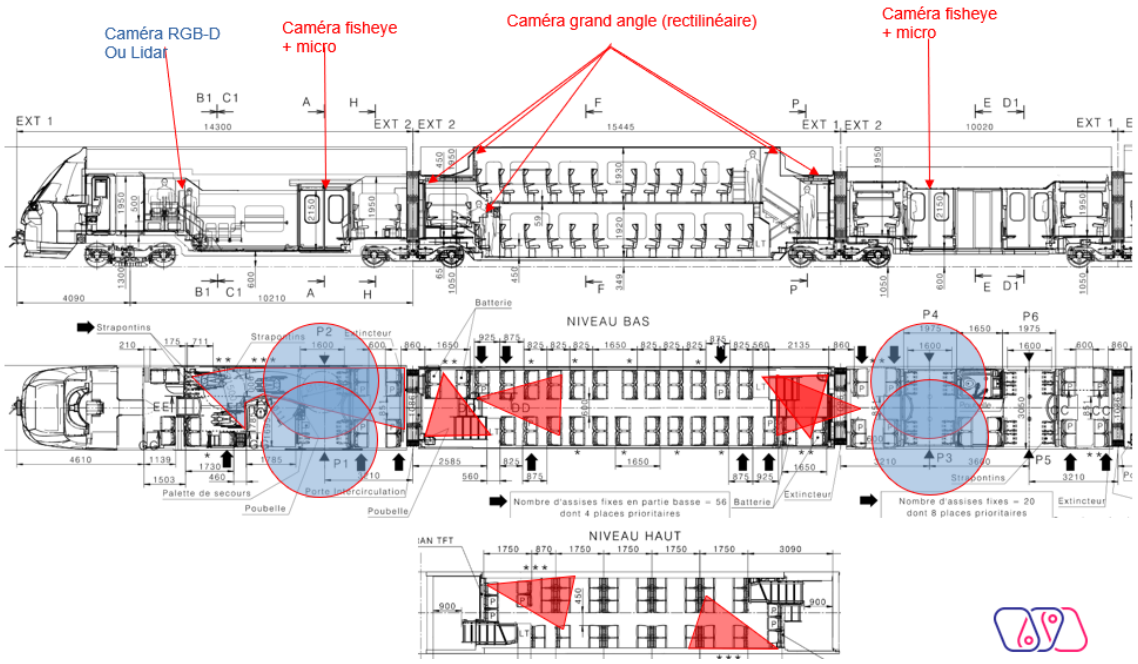


FIGURE 5.4 – Positionnement des caméras dans une voiture passager d'un train de type Regio2N.

de sièges, surveillant l'intérieur de la voiture et plus particulièrement les rangées de sièges passagers. Les caméras surveillant l'intérieur sont de modèle Brickcom MD-300Np Star. Elles ont un champ de vision horizontal de 76°, vertical de 56°, une résolution de 2048 × 1536 à 30 frames par seconde. Ce sont des caméras grand angle qui donnent des images rectilinéaires. Les caméras aux portes de la voiture sont de modèle Strabag CAM MD360S M12-B, avec objectif fisheye, ce qui leur confère un champ de vision horizontal et vertical de 182°, pour une résolution de 2048 × 1536 à 30 frames par seconde.

5.2.2 Scénarios à l'étude

Dans le cadre du programme Train Autonome Service Voyageurs et en particulier le lot 15 (relation avec les voyageurs), différents scénarios sont envisagés et relèvent des situations standards ou complexes décrites ci-après. Tous vérifient les deux propriétés suivantes :

- Les scénarios commencent par l'entrée des usagers dans la voiture et se terminent par leur sortie de la voiture ;
- L'entrée des usagers dans la voiture peut se faire depuis l'extérieur (portes automatiques) ou depuis une autre voiture.

(a) Caméra fisheye (b) Caméra grand angle (rectilinéaire)

FIGURE 5.5 – Deux types de caméras utilisées dans le prototype de train autonome.

Situations standards

Les situations standards sont des scénarios courants à bord des trains que les algorithmes doivent être en mesure de gérer sans difficulté. Voici quelques exemples de scénarios considérés :

- *Entrée-sortie de la voiture sans s'asseoir* : l'usager a réservé une place dans une voiture différente de celle filmée. Il entre dans la voiture et se dirige vers la sortie de la voiture sans s'asseoir. Chaque usager participant passe l'un après l'autre, un seul étant visible à la fois ;
- *Entrée-sortie de la voiture en s'asseyant avec changement de place (Entrée + Assis + Changement de place + Sortie)* : l'usager entre, cherche éventuellement sa place, potentiellement avec son téléphone portable ou le ticket de train dans la main pour vérifier son numéro de place. L'usager s'assoit à une place, et change éventuellement plusieurs fois de place. Enfin, l'usager se lève et se dirige vers la sortie de la voiture.

Situations complexes

Les situations complexes sont des scénarios peu courants à bord des trains mais dont la dangerosité éventuelle rend leur traitement par les algorithmes important. Celles-ci incluent les scenarios suivants :

- *Evanouissement sans aide* : l'usager montre des signes de malaise, et passe d'une posture debout à une posture allongée. Les autres usagers l'ignorent ;
- *Evanouissement avec aide* : l'usager montre des signes de malaise, et passe d'une posture debout à une posture allongée. Il est aidé par un usager et les autres s'attroupent autour de lui ;

- *Harcèlement* : un usager envahit l'espace d'un autre usager, avec potentiellement contact physique (sans arme), et le gêne ainsi pendant un temps prolongé ;
- *Altercation* : deux usagers se croisent, se cognent en se croisant, s'invectivent, s'empoignent, puis tombent au sol. L'un finit par l'emporter et sort de la voiture. L'autre finit par se relever et sort également de la voiture.

Organisation de l'enregistrement des scénarios

L'enregistrement des scénarios s'est déroulé pendant une après-midi passée au technicentre SNCF de Lille. Nous avons organisé cet enregistrement à l'aide de documents d'organisation illustrés dans les figures 5.6 à 5.8. Pour les scénarios où les acteurs traversent le train les uns à la suite des autres, un code couleur permet d'identifier chaque acteur et ainsi de comprendre rapidement l'ordre de passage dans la voiture. Quand une personne entre, elle barre la case appropriée, et en dessous de cette case est indiqué le temps que la personne suivante doit attendre avant de pouvoir entrer. Pour les situations particulières, indiquées dans la figure 5.8, nous avons tiré au sort des nombres correspondant chacun à une situation particulière et noté le résultat dans les cases blanches.

Traversée simple									
P1	P2	P3	P4	P5	P2	P3	P4	P5	P1
	30	30	30	30	15	15	15	15	15

FIGURE 5.6 – Feuille d'organisation de l'acquisition de données vidéos : timing.

Evanouissement sans aide				
Prise 1	Personne qui s'évanouit :	George		
Prise 2	Personne qui s'évanouit :	Leo		
Evanouissement avec aide				
Prise 1	Personne qui s'évanouit :	Arnaud	Personne qui aide :	Leo
Prise 2	Personne qui s'évanouit :	Maxime	Personne qui aide :	George

FIGURE 5.7 – Feuille d'organisation de l'acquisition de données vidéos : scénario.

Harcèlement				
Prise 1	Harceleur :	George	Harcelé :	Laure
Prise 2	Harceleur :	Arnaud	Harcelé :	Maxime
Altercation				
Prise 1	Agresseur :	Sarah	Victime :	Leo
Prise 2	Agresseur :	Maxime	Victime :	George

FIGURE 5.8 – Feuille d'organisation de l'acquisition de données vidéos : acteur.

5.3 Détection de personnes aux portes du futur Train Autonome

Comme indiqué en section 5.2.1, les caméras aux portes du train Regio2N sont de type fisheye. Nous avons annoté pour la segmentation d'instance de personnes, une base de données de 137 images enregistrées aux portes. Cette base de données comporte des scénarios anormaux joués par des acteurs, afin d'évaluer les performances des approches de réseaux de neurones présentées dans le chapitre 3. Cette nouvelle base de données est appelée fisheye_Regio2N. Les résultats quantitatifs et qualitatifs sont respectivement présentés dans le tableau 5.1 et les figures 5.9 à 5.15. Nous pouvons remarquer que la référence1 (Mask R-CNN pré-entraîné sur MS-COCO) obtient les performances les plus faibles (APall de 29.3 contre 40.8, 40.4 et 42.9), ce qui montre à nouveau l'intérêt d'un affinage sur des données augmentées. De plus, l'augmentation de données basée sur un ensemble de 8 transformations FE obtient des résultats très proches de ceux obtenus avec un ensemble de 35 transformations FE (APall de 40.4 contre 40.8), ce qui confirme l'hypothèse formulée dans le Chapitre 3, qu'un ensemble d'augmentations simples fonctionne aussi bien qu'un ensemble d'augmentations complexes. Cependant, l'augmentation par renversement vertical (halfvflip, présente sous référence2) suffit pour obtenir de meilleurs résultats qu'avec les 2 types d'augmentations fisheye (APall de 42.9 contre 40.4 et 40.8). Ceci est probablement dû au fait que dans cette base de données, les personnes présentes ne se trouvent pas aux extrémités de l'image, ce qui fait que leur apparence est moins déformée par l'effet fisheye. Enfin, nous pouvons mentionner que le modèle n'a pas été affiné sur des données issues du Regio2N, et que nous obtenons néanmoins de bonnes performances même sur des données très différentes de celles sur lesquelles le modèle a été entraîné.

Le modèle a également été testé sur une base de données d'images rectilinéaires

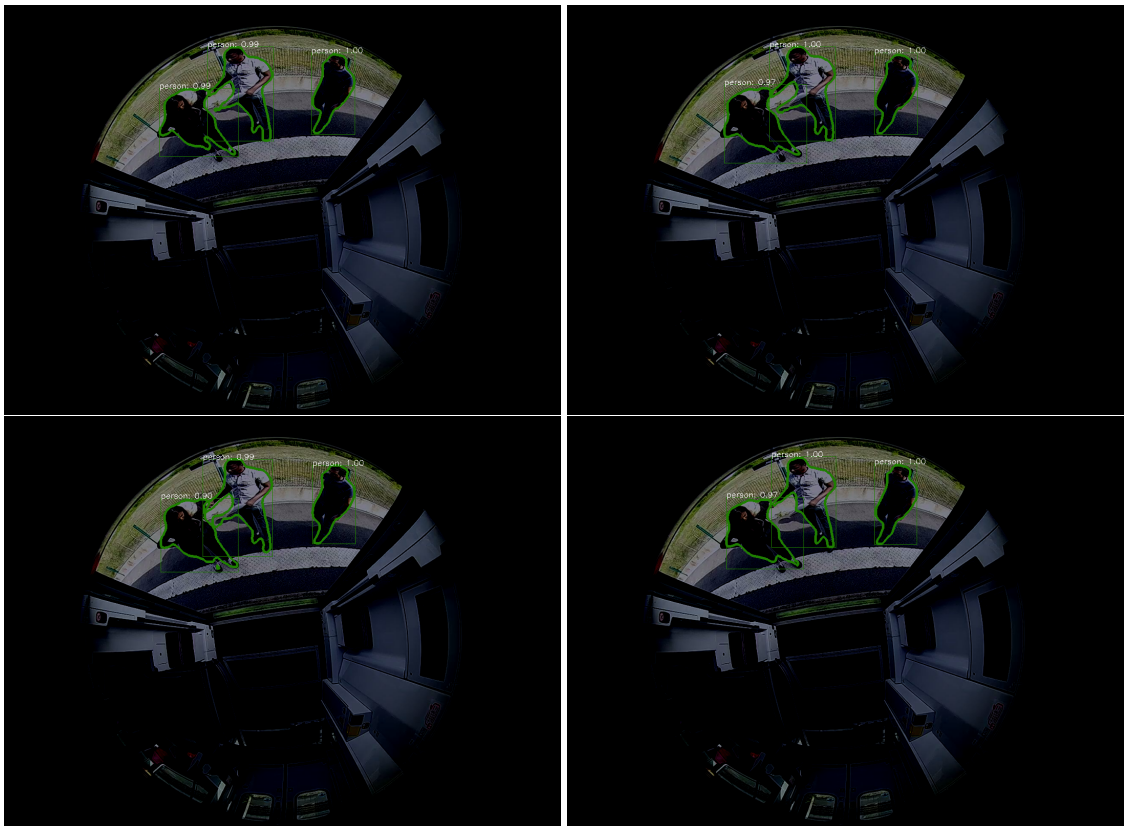


FIGURE 5.9 – Exemples de résultats qualitatifs sur la base de donnée fisheye_Regio2N. En haut à gauche : référence1 (Mask R-CNN pré-entraîné sur MS-COCO) ; haut à droite : 35FE transformations ; bas à gauche : 8FE transformations ; bas à droite : référence2 (Mask R-CNN affiné avec halfvflip).

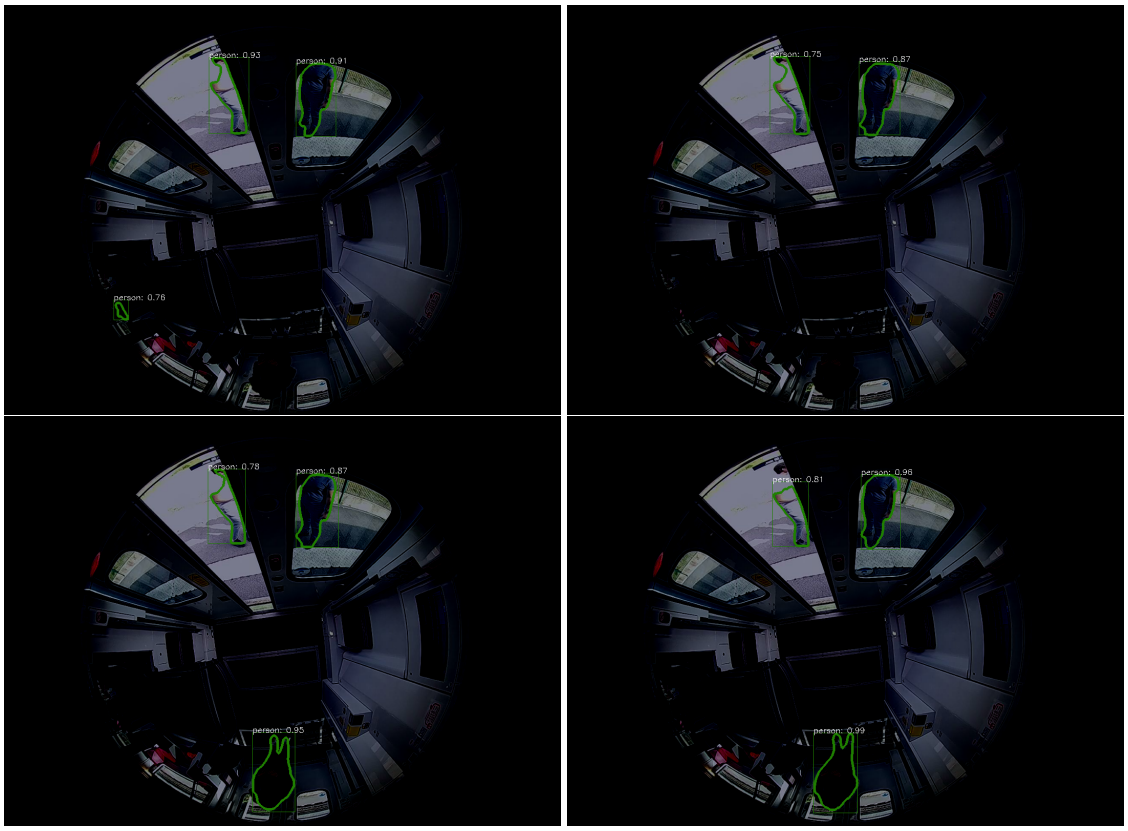


FIGURE 5.10 – Exemples de résultats qualitatifs sur la base de donnée fisheye_Regio2N. En haut à gauche : référence1 (Mask R-CNN pré-entraîné sur MS-COCO) ; haut à droite : 35FE transformations ; bas à gauche : 8FE transformations ; bas à droite : référence2 (Mask R-CNN affiné avec halfvflip).

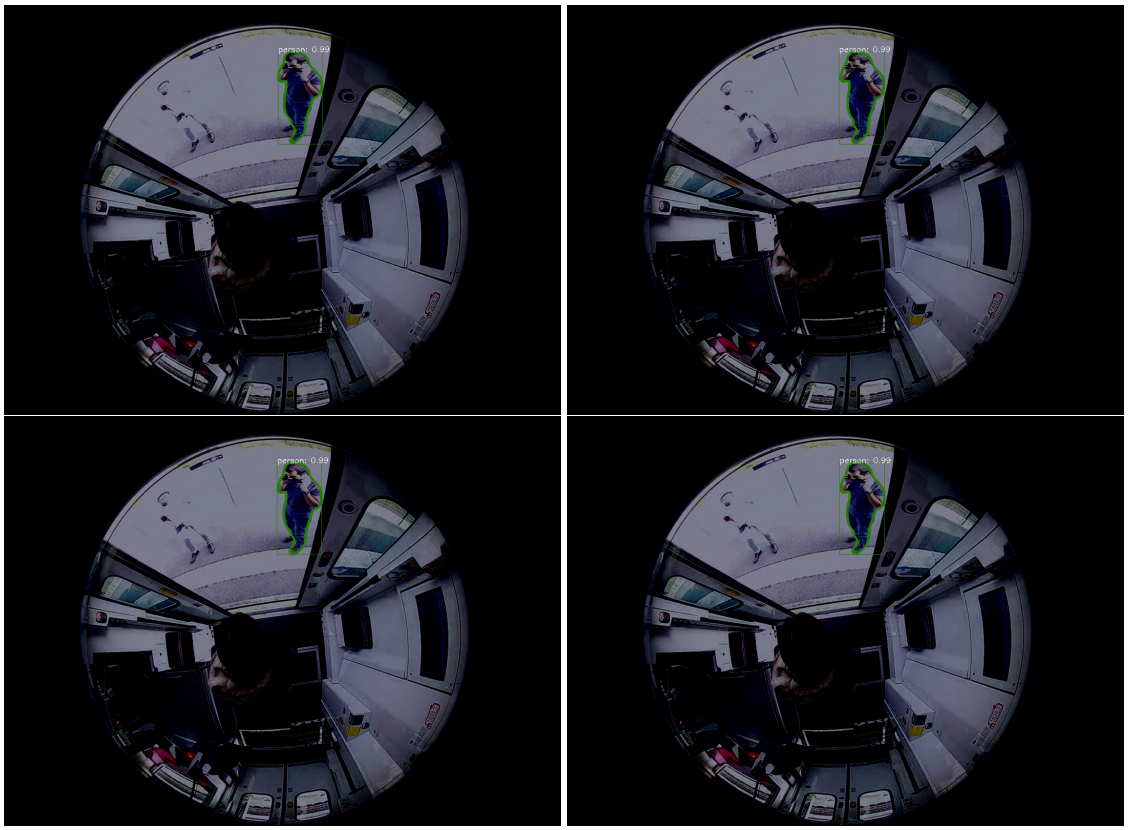


FIGURE 5.11 – Exemples de résultats qualitatifs sur la base de donnée fisheye_Regio2N. En haut à gauche : référence1 (Mask R-CNN pré-entraîné sur MS-COCO) ; haut à droite : 35FE transformations ; bas à gauche : 8FE transformations ; bas à droite : référence2 (Mask R-CNN affiné avec halfvflip).

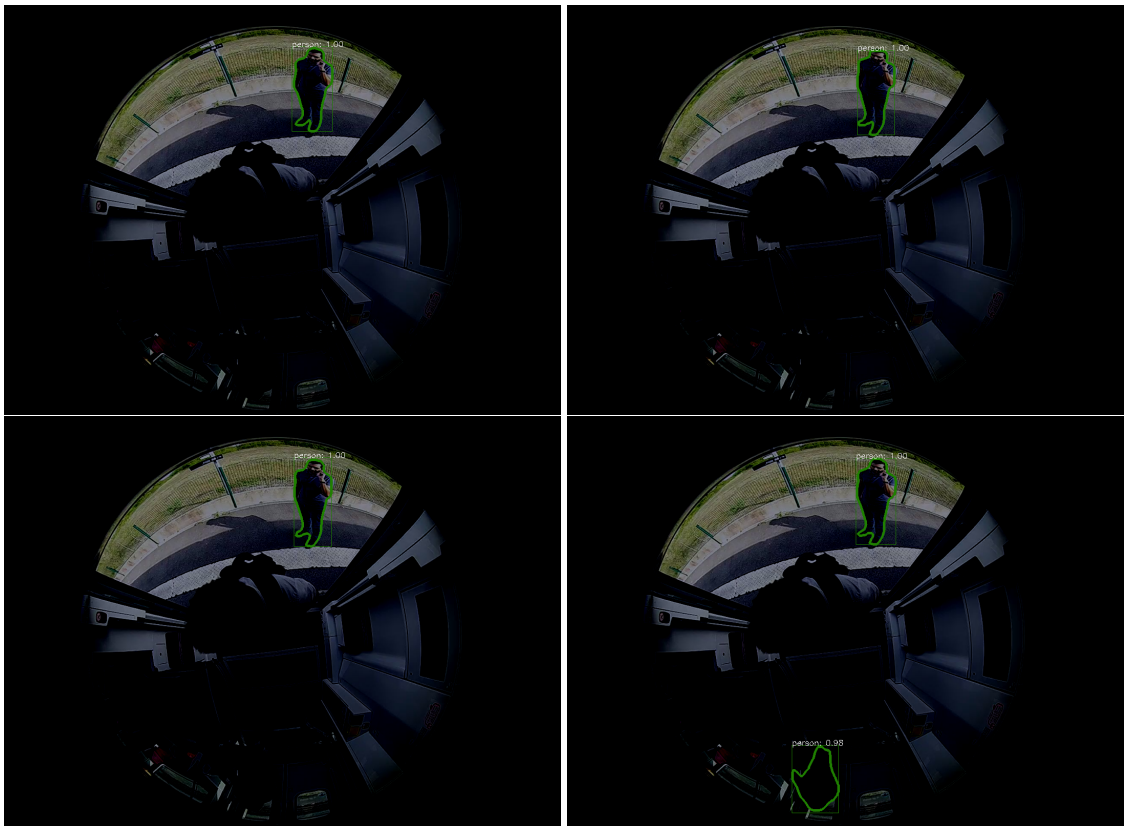


FIGURE 5.12 – Exemples de résultats qualitatifs sur la base de donnée fisheye_Regio2N. En haut à gauche : référence1 (Mask R-CNN pré-entraîné sur MS-COCO) ; haut à droite : 35FE transformations ; bas à gauche : 8FE transformations ; bas à droite : référence2 (Mask R-CNN affiné avec halfvflip).

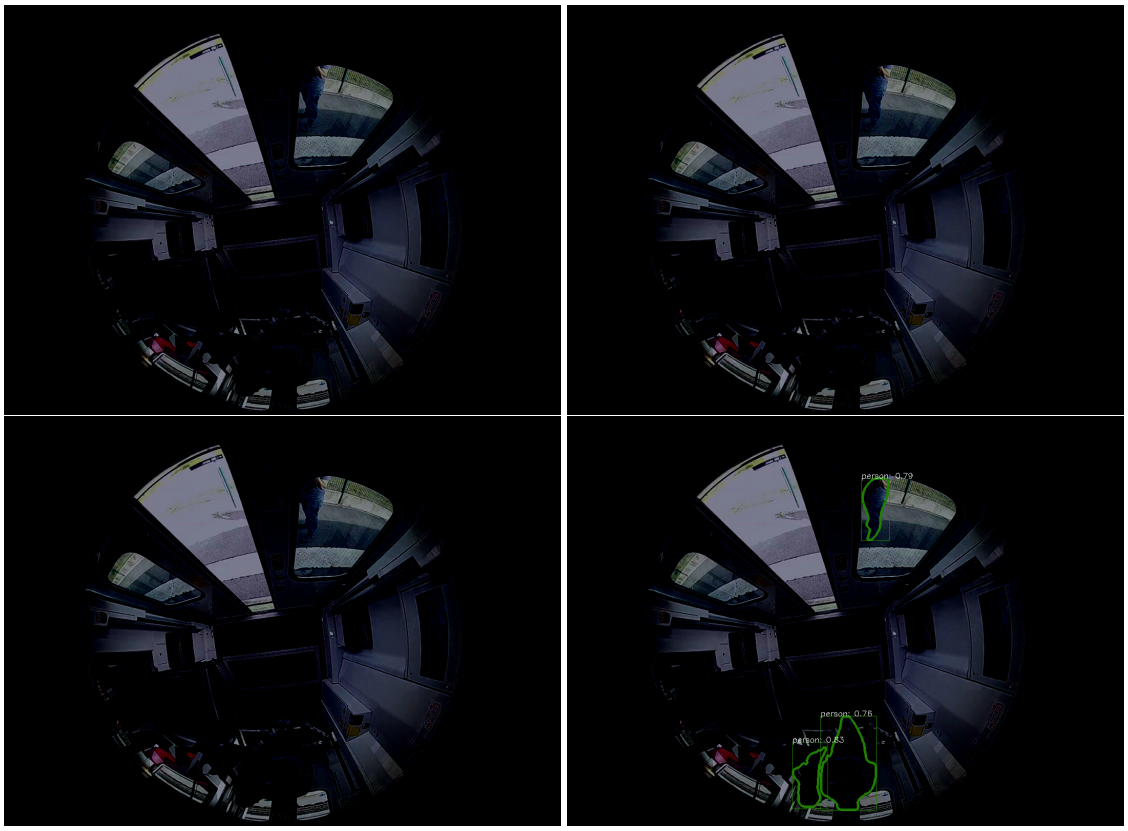


FIGURE 5.13 – Exemples de résultats qualitatifs sur la base de donnée fisheye_Regio2N. En haut à gauche : référence1 (Mask R-CNN pré-entraîné sur MS-COCO) ; haut à droite : 35FE transformations ; bas à gauche : 8FE transformations ; bas à droite : référence2 (Mask R-CNN affiné avec halfvflip).

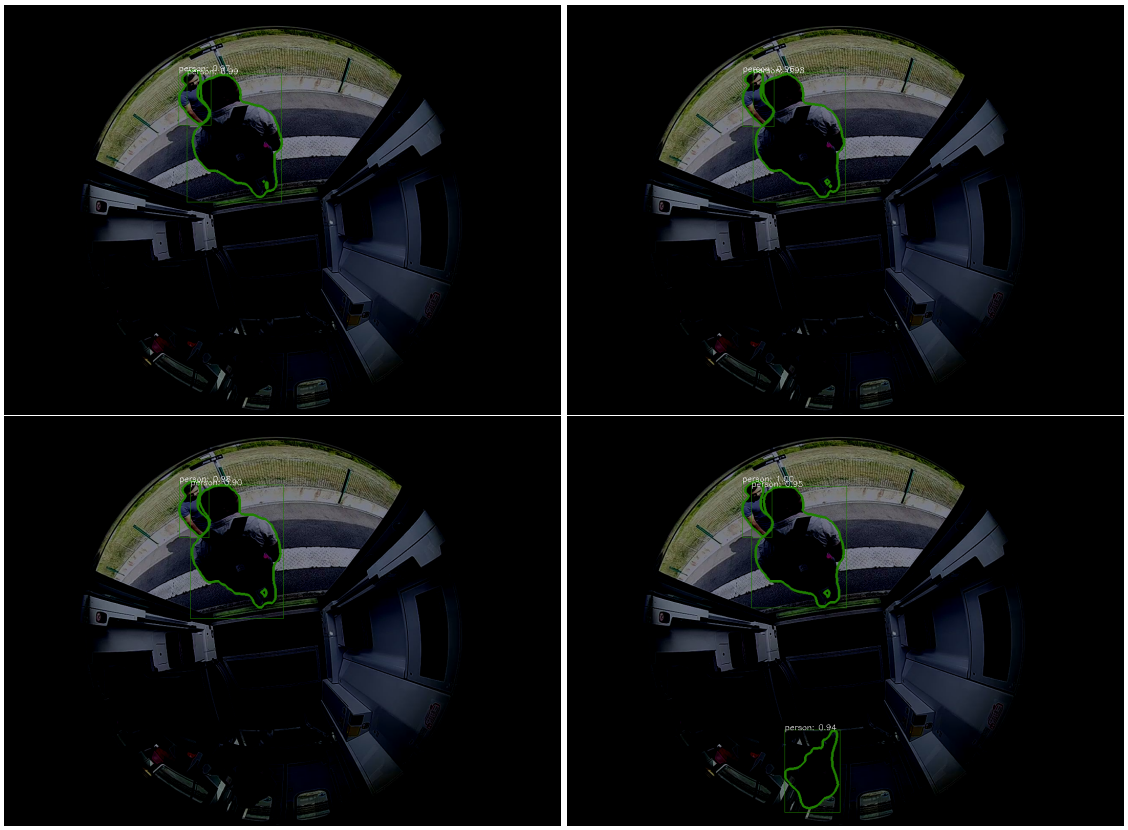


FIGURE 5.14 – Exemples de résultats qualitatifs sur la base de donnée fisheye_Regio2N. En haut à gauche : référence1 (Mask R-CNN pré-entraîné sur MS-COCO) ; haut à droite : 35FE transformations ; bas à gauche : 8FE transformations ; bas à droite : référence2 (Mask R-CNN affiné avec halfvflip).

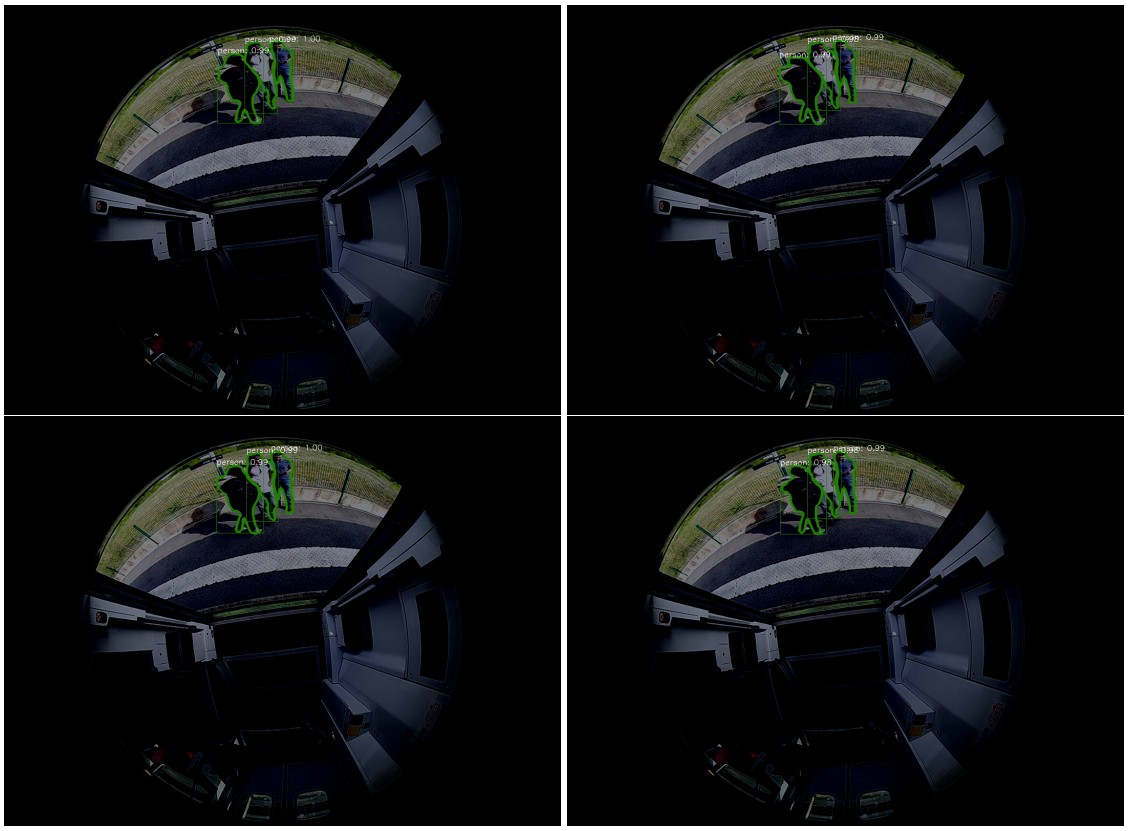


FIGURE 5.15 – Exemples de résultats qualitatifs sur la base de donnée fisheye_Regio2N. En haut à gauche : référence1 (Mask R-CNN pré-entraîné sur MS-COCO) ; haut à droite : 35FE transformations ; bas à gauche : 8FE transformations ; bas à droite : référence2 (Mask R-CNN affiné avec halfvflip).

	APall	**AP50**	**AP75**	**APM**	**APL**
Réference1 (Mask R-CNN pré-entraîné sur MS-COCO)	29.3	42.6	35.0	32.1	37.4
35FE transformations	40.8	58.9	**49.7**	**36.3**	50.0
8FE transformations	40.4	60.6	49.4	33.8	50.6
Référence2 (Mask R-CNN affiné avec halfvflip)	**42.9**	**64.0**	49.3	34.1	**52.0**

Tableau 5.1 – Performance d'un Mask R-CNN ResNet-50 entraîné avec 50% d'augmentation FE (avec deux ensembles de transformations différents) vs poids originaux de référence et affinés avec halfvflip.

enregistrées dans un train Regio2N et présentant des scénarios détaillés dans la sous-section 5.2.2. Les résultats sont illustrés dans les figures 5.16 à 5.18 et permettent de montrer qualitativement que les modèles entraînés avec une augmentation de données fisheye (pour 50% de leurs exemples d'entraînement) maintiennent des performances acceptables sur des images rectilinéaires.

5.4 Détection de points clés des squelettes des usagers du futur Train Autonome.

Afin d'évaluer l'algorithme de détection de points clés de squelettes multi-personnes proposé dans le chapitre 4, non plus sur des bases de données existantes dans la littérature, mais sur des données réelles, deux séquences vidéos comportant des scénarios acquis dans un train Regio2N et détaillés dans la sous-section 5.2.2 sont utilisées.

- La première séquence concerne le scénario 1 : *entrée-sortie de la voiture sans s'asseoir*. Une annotation est réalisée toutes les 10 frames, soit un total de 44 frames annotées ;
- la deuxième séquence concerne le scénario 2 : *entrée-sortie de la voiture en s'asseyant avec changement de place*. Une annotation est également réalisée toutes les 10 frames, soit un total de 45 frames annotées.

Deux exemples d'annotations correspondant respectivement aux scénarios 1 et 2 sont présentés dans les figures 5.19 et 5.20. L'annotation a été réalisée en notant pour chaque partie du corps les coordonnées (x, y) de la même manière que celle présentée dans la sous-section 4.3.3 et la section 4.5.

Nous effectuons ces expériences avec la meilleure configuration observée dans le chapitre 4, et avec un cercle de 10 pixels de rayon autour de chaque point clé de vérité terrain pour compter les vrais positifs. De plus, nous expérimentons deux nouvelles options d'entraînement. La première consiste à cacher au hasard 30% des points clés du

FIGURE 5.16 – Exemples de résultats qualitatifs obtenus pour le scénario 1 d'une base de données d'images rectilinéaires. En haut à gauche : référence1 (Mask R-CNN pré-entraîné sur MS-COCO) ; haut à droite : 35FE transformations ; bas à gauche : 8FE transformations ; bas à droite : référence2 (Mask R-CNN affiné avec halfvflip).

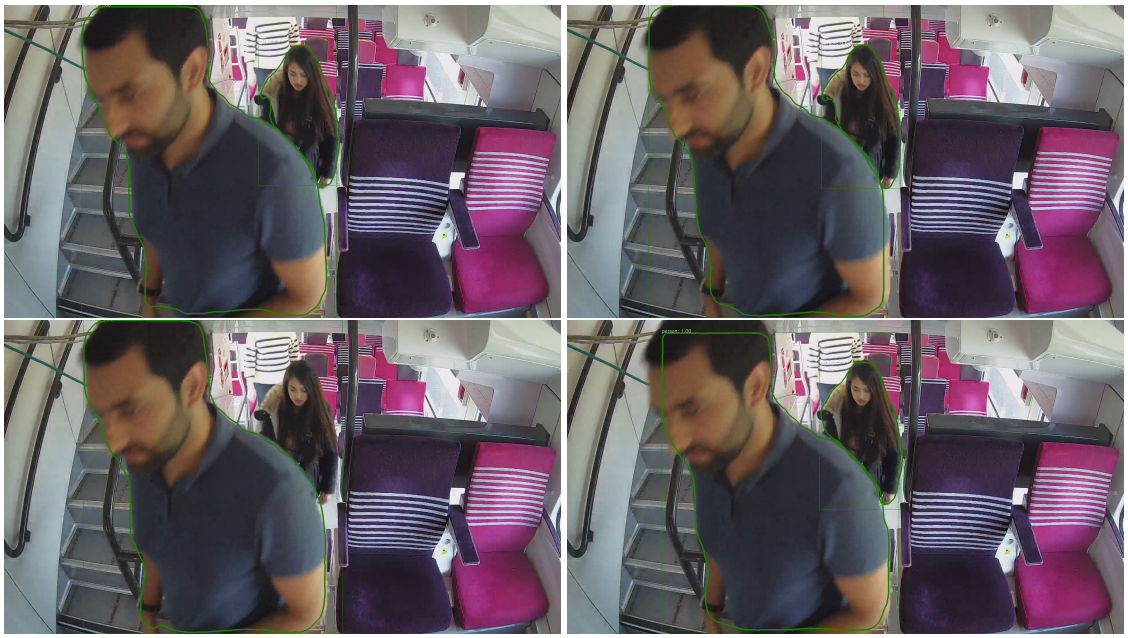


FIGURE 5.17 – Exemples de résultats qualitatifs obtenus pour le scénario 1 d'une base de données d'images rectilinéaires. En haut à gauche : référence1 (Mask R-CNN pré-entraîné sur MS-COCO) ; haut à droite : 35FE transformations ; bas à gauche : 8FE transformations ; bas à droite : référence2 (Mask R-CNN affiné avec halfvflip).

FIGURE 5.18 – Exemples de résultats qualitatifs obtenus pour le scénario 1 d'une base de données d'images rectilinéaires. En haut à gauche : référence1 (Mask R-CNN pré-entraîné sur MS-COCO) ; haut à droite : 35FE transformations ; bas à gauche : 8FE transformations ; bas à droite : référence2 (Mask R-CNN affiné avec halfvflip).

FIGURE 5.19 – Exemple d'annotation vérité terrain pour le scénario 1.

FIGURE 5.20 – Exemple d'annotation vérité terrain pour le scénario 2.

corps pendant la phase de mémorisation. La deuxième consiste à cacher tous les squelettes pendant la phase de mémorisation dans 30% des exemples d'entraînement, choisis au hasard. Ces deux options ont pour objectif d'améliorer les capacités de STMskeletons à détecter de nouveaux squelettes et points clés, ce qui serait très utile en conditions réelles d'exploitation. STMskeletons n'est pas affiné sur ces nouvelles données car elles sont en nombre trop réduit pour le permettre. Les résultats quantitatifs sont présentés dans le tableau 5.2. Les résultats sont illustrés qualitativement dans les figures 5.21 et 5.22. Nous remarquons, sans nouvelles options d'entraînement, une précision satisfaisante de 64.0 sur le scénario 1 et de 70.5 sur le scénario 2. Le rappel est quant à lui plus faible, de 11.7 sur le scénario 1 et de 21.5 sur le scénario 2. Ceci indique que l'algorithme produit peu de faux positifs mais échoue à détecter un grand nombre de points. Les deux nouvelles options d'entraînement, dont les résultats sont indiqués en deuxième partie du tableau, ne semblent pas améliorer les performances de manière significative, puisque le F1-score avec nouvelles options d'entraînement est plus élevé sur le scénario 1 que sans nouvelles options d'entraînement mais est en deçà sur le scénario 2. En général, le haut du corps semble être bien détecté, par rapport au bas du corps (hanches, genoux ...) que l'algorithme peine à détecter. Pour pallier à ces lacunes, il faudrait un grand nombre de données étiquetées supplémentaires dans des contextes similaires à ceux du Regio2N. Une autre manière pourrait être de combler les lacunes de détection en effectuant un suivi temporel.

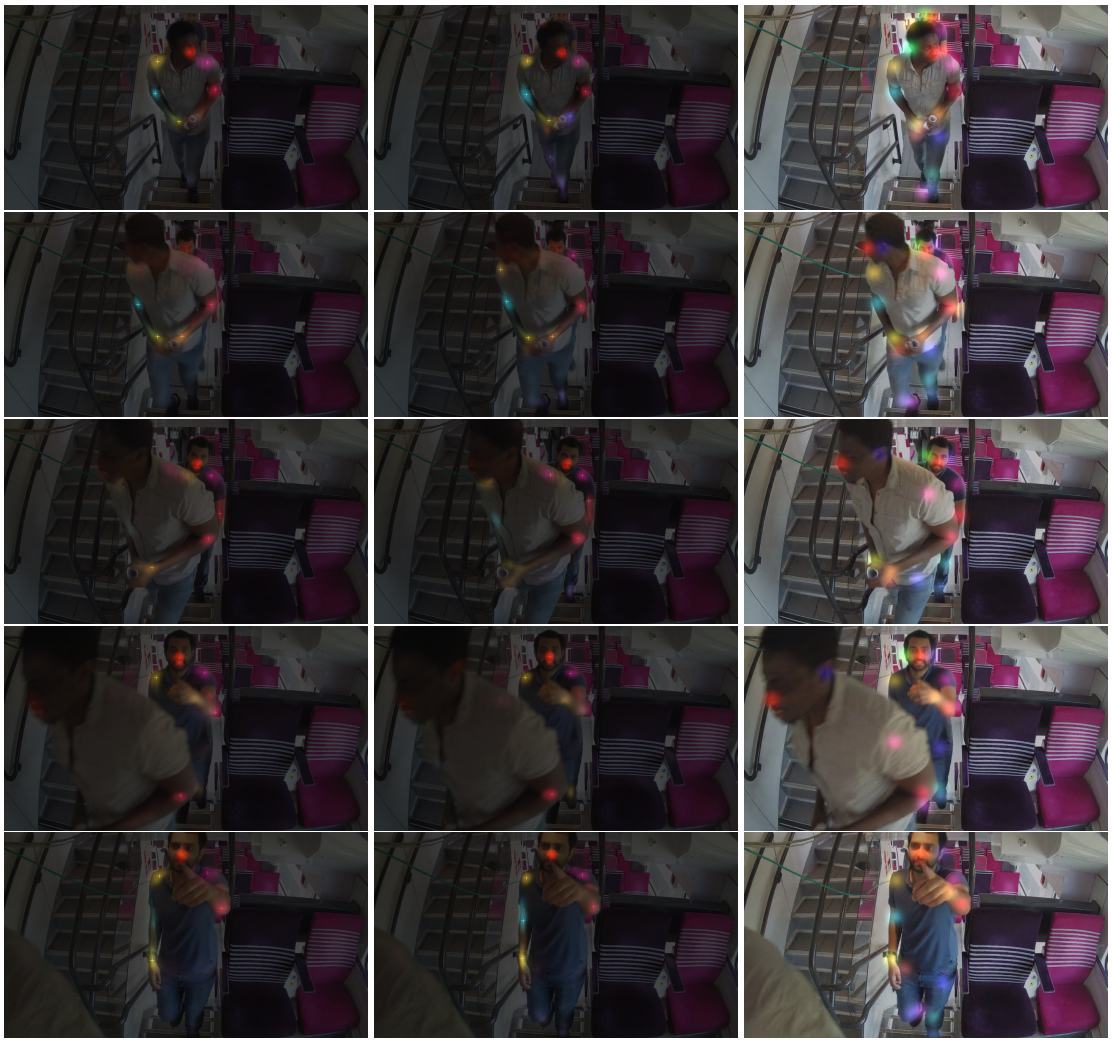


FIGURE 5.21 – Illustration des résultats de l'algorithme STMskeletons obtenus sur une vidéo acquise à l'intérieur d'une voiture de train de type Regio2N. Gauche : prédiction par STMskeletons. Milieu : prédiction par STMskeletons avec les nouvelles options d'entraînement. Droite : vérité terrain.

FIGURE 5.22 – Illustration des résultats de l'algorithme STMskeletons obtenus sur une vidéo acquise à l'intérieur d'une voiture de train de type Regio2N. Gauche : prédiction par STMskeletons. Milieu : prédiction par STMskeletons avec les nouvelles options d'entraînement. Droite : vérité terrain.

	Précision	Rappel	F1-score
SANS nouvelles options d'entraînement			
scénario 1	64.0	11.7	19.8
scénario 2	70.5	21.5	33.0
AVEC nouvelles options d'entraînement			
scénario 1	61.6	13.6	22.3
scénario 2	63.8	20.0	30.5

Tableau 5.2 – Evaluation quantitative de l'algorithme STMskeletons sur les deux scénarios présentés.

5.5 Conclusion

Dans ce chapitre, nous avons cherché à appliquer et évaluer les deux approches développées dans les chapitres 3 et 4 sur des datasets réels collectés dans un type de train Regio2N qui sera utilisé dans le cadre du futur train autonome. L'acquisition des données images a été effectuée à l'aide de deux modèles de caméras différents, l'une à objectif fisheye (présentant des distorsions en barillet), placée au dessus des portes, l'autre à objectif grand angle (présentant des images rectilinéaires), placée à l'intérieur du train et observant une voiture de voyageurs. Le dataset collecté par la caméra fisheye a servi à évaluer les différentes configurations d'affinage présentées dans le chapitre 3. Cela a permis de confirmer l'utilité de l'affinage avec une augmentation de données sur des images fisheye, puisque la référence obtient un APall de 29.3 contre 40.8 avec 35 FE transformations. Cependant les résultats sont mitigés car l'augmentation de données par renversement vertical obtient un APall de 42.9 donc légèrement supérieur. Le dataset collecté par la caméra grand angle a servi à évaluer la meilleure configuration obtenue dans le chapitre 4, ainsi que de nouvelles options d'entraînement conçues pour faciliter la détection de points clés de squelettes de personnes. Les résultats ont une précision satisfaisante mais obtiennent un rappel faible, et les nouvelles options d'entraînement ne semblent pas apporter de bénéfice significatif.

Chapitre 6

Conclusion et perspectives

6.1 Conclusion et contributions

Dans cette thèse, nous avons cherché à adapter d'une part, un algorithme de segmentation d'instance pour des images fisheye et d'autre part, adapter un algorithme de Video Object Segmentation pour la détection de points clés de squelettes de personnes dans des vidéos. Ces propositions ont été testées et validées sur des bases de données bien connues de la littérature mais aussi sur deux bases de données acquises dans le cadre applicatif du programme TASV (Train Autonome Service Voyageur) impliquant l'IRT Railenium. Ce programme vise à produire un train autonome de type GoA4 (Grade of Automation 4), sans personnel à bord, à l'horizon 2023 pour une industrialisation à partir de 2025, et ainsi rendre le trafic plus fluide et permettre une consommation d'énergie plus faible.

Après avoir présenté un état de l'art, nous avons proposé, dans le chapitre 3, une méthode dont l'objectif est de permettre l'usage du détecteur de segmentation d'instance Mask R-CNN sur des images de type fisheye, tout en conservant de bonnes performances sur des images sans distorsion (rectilinéaires). Ceci s'avère intéressant dans le cadre du projet TASV où ces deux types de caméras sont utilisées. Le manque de base de données réelles fisheye fait qu'il est actuellement impossible d'entraîner Mask R-CNN sur des images fisheye réelles. Nous avons donc proposé une méthode d'entraînement utilisant des images rectilinéaires transformées en images fisheye synthétiques (FE) de manière à ce qu'elles présentent le même type de distorsion. Pour ce faire, deux ensembles de transformations FE ont été conçus, le premier contient 35 transformations complexes, et le deuxième 8 transformations simplifiées. Pendant l'affinage, le modèle Mask R-CNN est exposé à la fois à des images rectilinéaires, et à la fois à des images augmentées FE. Une première expérience a permis de conclure qu'affiner un backbone préentraîné sur des images rectilinéaires donne de meilleurs résultats (APall de 0.16) sur les images FE que de partir d'un backbone initialisé aléatoirement (APall de 0.12). Une deuxième expérience comparant différents ratios d'augmentation FE a permis de décider qu'un ratio de 50% est un bon compromis entre les performances rectilinéaires et

les performances fisheye. En effet nous obtenons une augmentation de 9 points en APall sur les performances fisheye et une diminution de seulement 2 points en APall sur les performances rectilinéaires. Des expériences ont ensuite été réalisées sur deux bases de données fisheye réelles (acquises dans le cadre du programme TASV), annotées pour la segmentation d'instance de personnes. Les résultats ont montré l'intérêt de l'augmentation utilisant 35 transformations FE sur des images fisheye réelles, avec notamment un gain de 16.5 en APall sur la base de données trainDoorAug. Des comparaisons ont ensuite été réalisées avec l'augmentation utilisant 8 transformations simplifiées et une augmentation par renversement vertical, et ont montré que l'usage de 35 transformations complexes n'est pas nécessaire, car la différence en termes de APall n'est que de un point par rapport à celui obtenu avec 8 transformations.

Dans le chapitre 4, nous avons proposé d'adapter l'algorithme de Video Object Segmentation basé sur STM à l'estimation et au suivi de points clés multi-personnes dans des vidéos. Plusieurs expériences ont été réalisées avec l'algorithme STM original afin de prouver la viabilité du concept, et montrer que l'algorithme est capable d'effectuer la segmentation de squelette dans des vidéos, d'abord avec une cible binaire (obtenant un F1-score de 71.7 sur PoseTrack18_Validation FULL), puis avec une carte de confiance (obtenant un F1-score de 37.2 sur PoseTrack18_Validation FULL), et enfin en étant entraîné à suivre chaque arrête du corps humain de manière individuelle (obtenant un F1-score de 30.1 sur PoseTrack18_Validation FULL). Nous avons ensuite modifié l'architecture de STM, au niveau des premières couches de l'encodeur et des dernières couches du décodeur, afin de le rendre adaptatif à l'estimation de cartes de confiance individuelles pour chaque point clé et arête du corps, ce qui mène à une nouvelle architecture que nous avons appelée STMskeletons. L'entraînement se déroule tout d'abord sur des données synthétiques, composées de courtes séquences vidéos créées à partir d'images de la base de données MS-COCO, puis sur des données vidéos réelles tirées de la base de données PoseTrack18. Le modèle entraîné avec affinage sur PoseTrack18 est comparé à une version sans affinage (F1-score de 47.9 contre 38.9), et permet de conclure que l'affinage est d'un grand intérêt pour les performances en conditions réelles d'exploitation. Différentes options d'augmentation de données ont ensuite été conçues, ainsi qu'un type d'entraînement appelé "cyclique" dans lequel le modèle réutilise à la frame N+1 des prédictions réalisées à la frame N. Une étude ablative comparant toutes ces options permet de conclure de l'intérêt de l'entraînement cyclique (F1-score de 46.8 contre 29.4), et des options d'entraînement "bait" et "jitter" (F1-score de 47.6 contre 46), mais de l'absence relative d'intérêt des options d'entraînement "rand" (F1-score de 47.6 contre 47.9) et "dull_clouds" (F1-score de 44 contre 47.6). Enfin, une expérience présentant la métrique de rappel au fil du temps montre que l'intérêt de l'entraînement cyclique et de l'affinage réside dans la capacité de suivi long terme plutôt que dans le court terme. En effet, dans les premières frames, les différences sur la métrique de rappel sont inférieures à 5%, et

augmentent au cours du temps.

Dans le chapitre 5, nous avons appliqué et évalué les méthodes développées dans les deux chapitres précédents dans le contexte applicatif du futur train autonome, à l'aide de deux bases de données réelles (acquises dans un train Regio2N durant cette thèse) et annotées, l'une pour la segmentation d'instance, l'autre pour l'estimation de points clés multi-personnes dans des vidéos. Pour la segmentation d'instance, nous comparons à nouveau la référence entraînée sur des images rectilinéaire du dataset MS-COCO à des réseaux affinés avec une augmentation de données comportant respectivement 35 et 8 transformations FE, et avec renversement vertical. Les résultats sont mitigés car l'augmentation de données par renversement vertical obtient de meilleurs résultats que les augmentations de données par transformations FE (un APall de 42.9 contre 40.4 avec 8 transformations FE et un APall de 40.8 avec 35 transformations FE). Ces résultats mitigés viennent probablement du fait de la faible distorsion des personnes présentes dans les images du dataset car situées au centre de l'image. Pour l'estimation des points clés multi-personnes dans des vidéos, nous réutilisons la meilleure configuration obtenue dans le chapitre 4, que nous entraînons également avec deux nouvelles options d'entraînement proposées pour améliorer la détection (et donc la métrique de rappel). Nous évaluons ces deux configurations sur deux scénarios, le scénario 1 étant un usager qui entre puis sort et le scénario 2 étant un usager qui entre dans la voiture, s'assoit, change de place, puis sort de la voiture. Les résultats sont satisfaisants mais les nouvelles options d'entraînement n'apportent pas d'amélioration significative de la métrique de rappel, avec un F1-score de 13.6 contre 11.7 pour le scénario 1 et un F1-score de 20.0 contre 21.5 pour le scénario 2.

6.2 Perspectives

6.2.1 Vers la reconnaissance des actions des usagers

Les perspectives du travail présenté dans cette thèse s'organisent autour des objectifs du lot 15 du programme Train Autonome indiqués en section 5.1.1. Nous nous intéressons donc principalement à l'amélioration des capacités d'analyse automatique des situations présentes à bord du train. Un objectif majeur d'amélioration des capacités d'analyse serait de permettre la reconnaissance d'actions. La reconnaissance d'actions à partir de squelettes de personnes est un domaine de recherche très actif, et serait donc une perspective intéressante du Chapitre 4. Le travail présenté dans ce chapitre s'arrête à la détection de points clés de squelettes multi-personnes dans des vidéos. Pour aller plus loin, il faudrait, après la détection des points clés de squelettes, les assembler dans chaque frame pour former des squelettes complets à l'aide des Part Affinity Maps que

nous estimons et qui permettent de calculer l'affinité entre des points clés. De plus, une étape de matching entre les frames, potentiellement inspirée des algorithmes existants de l'état de l'art, permettrait d'effectuer le suivi de ces squelettes dans le temps, accomplissant ainsi la tâche de suivi de pose. Cela permettrait de pouvoir réaliser une comparaison avec les algorithmes existants de suivi de pose, ce qui manque actuellement au chapitre 4, mais aussi de permettre l'usage d'algorithmes existant de reconnaissance d'action à partir de squelettes. Nous pensons aussi qu'un algorithme de détection de pose réalisé toutes les x frames performant pourrait être couplé avec celui de suivi de pose, afin d'améliorer les performances globales.

6.2.2 Nouvelles bases de données

Pour l'application à des cas d'utilisation réels dans le contexte du futur train autonome, la collection et l'annotation de nouvelles bases de données en segmentation et en suivi de pose seraient de futures étapes importantes. Il serait par ailleurs utile de considérer deux types de capteurs : une caméra fisheye et une caméra rectilinéaire. Ces nouvelles bases de données devraient être idéalement de plus grande taille, afin de permettre à la fois, une mesure plus fine des performances des algorithmes, mais aussi un meilleur entraînement par affinage pour améliorer les performances réelles. Pour arriver à ces objectifs, il faudrait compter plusieurs milliers d'images annotées, divisées en courtes séquences de quelques secondes chacune. De plus, ces bases de données devront contenir une grande diversité de scénarios (standards comme complexes), de poses de personnes, d'usagers, et un grand nombre de situations difficiles à traiter comme des occlusions, des mouvements rapides et complexes, des foules denses, etc.

6.2.3 Des algorithmes plus robustes

Une autre perspective intéressante à ces travaux serait de chercher à améliorer la rapidité des algorithmes proposés afin de pouvoir traiter le flux vidéo de surveillance en temps réel, ce qui est crucial pour une application réelle. Poursuivre la modification de l'architecture de STMskeletons pourrait permettre d'améliorer la rapidité et/ou les performances de détection en conditions réelles, et en particulier, les performances en termes de rappel qui étaient mitigées dans la section 5.4. Pour ce faire, il serait possible de s'inspirer des algorithmes existants d'estimation de pose bottom-up tel que OpenPose. Comme perspective du chapitre 3, nous pouvons aussi expérimenter avec des transformations fisheye utilisant des paramètres qui correspondent à ceux d'une "vraie" caméra fisheye, afin de voir si les performances en sont améliorées. De telles expériences ont déjà été réalisées dans [Blott *et al.*, 2018] et ont donné des résultats positifs, et nous pensons que cela serait bénéfique dans notre contexte réel si l'on parvenait à les repro-

duire. Une dernière perspective serait éventuellement d'expérimenter l'adaptation de l'architecture de Mask R-CNN aux images fisheye. Par exemple, il serait possible d'utiliser des représentations de boîtes englobantes orientées, et donc davantage adaptées aux différentes orientations que peuvent prendre les objets dans les images fisheye.

Bibliographie

[Andriluka *et al.*, 2018] Andriluka, M., Iqbal, U., Ensafutdinov, E., Pishchulin, L., Milan, A., Gall, J. et B., S. (2018). PoseTrack : A benchmark for human pose estimation and tracking. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Andriluka *et al.*, 2014] Andriluka, M., Pishchulin, L., Gehler, P. et Schiele, B. (2014). 2d human pose estimation : New benchmark and state of the art analysis. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Andriluka *et al.*, 2009] Andriluka, M., Roth, S. et Schiele, B. (2009). Pictorial structures revisited : People detection and articulated pose estimation. *In IEEE Conference on Computer Vision and Pattern Recognition*, pages 1014–1021.

[Bao *et al.*, 2018] Bao, L., Wu, B. et Liu, W. (2018). Cnn in mrf : Video object segmentation via inference in a cnn-based higher-order spatio-temporal mrf. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5977–5986.

[Baumberg et Hogg, 1994] Baumberg, A. et Hogg, D. C. (1994). Learning flexible models from image sequences. *In European Conference on Computer Vision (ECCV)*.

[Belagiannis et Zisserman, 2017] Belagiannis, V. et Zisserman, A. (2017). Recurrent human pose estimation. *IEEE International Conference on Automatic Face and Gesture Recognition (FG)*, pages 468–475.

[Blott *et al.*, 2018] Blott, G., Takami, M. et Heipke, C. (2018). Semantic segmentation of fisheye images. *In European Conference on Computer Vision (ECCV)*, pages 181–196. Springer.

[Boser *et al.*, 1992] Boser, B. E., Guyon, I. M. et Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. *In Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, page 144–152, New York, NY, USA. Association for Computing Machinery.

[Bruckert *et al.*, 2021] Bruckert, A., Tavakoli, H. R., Liu, Z., Christie, M. et Meur, O. L. (2021). Deep saliency models : The quest for the loss function. *Neurocomputing*, 453: 693–704.

[Caelles *et al.*, 2017] Caelles, S., Maninis, K.-K., Pont-Tuset, J., Leal-Taixé, L., Cremers, D. et Gool, L. V. (2017). One-shot video object segmentation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5320–5329.

[Cao *et al.*, 2019] Cao, Z., Hidalgo Martinez, G., Simon, T., Wei, S. et Sheikh, Y. A. (2019). Openpose : Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[Carreira *et al.*, 2016] Carreira, J., Agrawal, P., Fragkiadaki, K. et Malik, J. (2016). Human pose estimation with iterative error feedback. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Caruso *et al.*, 2015] Caruso, D., Engel, J. et Cremers, D. (2015). Large-scale direct slam for omnidirectional cameras. *In 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 141–148.

[Chang *et al.*, 2013] Chang, J., Wei, D. et Fisher III, J. W. (2013). A video representation using temporal superpixels. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2051–2058.

[Chen *et al.*, 2018] Chen, Y., Pont-Tuset, J., Montes, A. et Gool, L. V. (2018). Blazingly fast video object segmentation with pixel-wise metric learning. *In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1189–1198.

[Christopher, 2009] Christopher, M. (2009). *Laser-augmented omnidirectional vision for 3D localisation and mapping*. Thèse de doctorat, École Nationale Supérieure des Mines de Paris.

[Ciresan *et al.*, 2011] Ciresan, D. C., Meier, U., Masci, J., Gambardella, L. M. et Schmidhuber, J. (2011). Flexible, high performance convolutional neural networks for image classification. *In International Joint Conference on Artificial Intelligence (IJCAI)*.

[Cognasse, 2018] Cognasse, O. (2018). https://www.usinenouvelle.com/article/la-sncf-fera-rouler-des-trains-autonomes-en-2023.N740669.

[Cordts *et al.*, 2016] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S. et Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Dalal et Triggs, 2005] Dalal, N. et Triggs, B. (2005). Histograms of oriented gradients for human detection. *In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893 vol. 1.

[Deng *et al.*, 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. et Fei-Fei, L. (2009). ImageNet : A large-scale hierarchical image database. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255.

[Doering *et al.*, 2018] Doering, A., Iqbal, U. et Gall, J. (2018). Joint flow : Temporal flow fields for multi person tracking. *In British Machine Vision Conference (BMVC)*.

[Duchi *et al.*, 2011] Duchi, J., Hazan, E. et Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159.

[Dufour *et al.*, 2021] Dufour, R., Meurie, C., Lézoray, O. et Mahtani, A. (2021). Segmentation d'instance dans des images fisheye. *In ORASIS*.

[Dufour *et al.*, 2022] Dufour, R., Meurie, C., Lézoray, O. et Mahtani, A. (2022). Space-time memory networks for multi-person skeleton body part detection. *In International Conference on Pattern Recognition and Artificial Intelligence (ICPRAI)*, volume 13364 de *Lecture Notes in Computer Science*, pages 78–90. Springer.

[Dufour *et al.*, 2020] Dufour, R., Meurie, C., Strauss, C. et Lézoray, O. (2020). Instance segmentation in fisheye images. *In International Conference on Image Processing Theory, Tools and Applications (IPTA)*, pages 1–6.

[Fan *et al.*, 2015] Fan, Q., Zhong, F., Lischinski, D., Cohen-Or, D. et Chen, B. (2015). Jumpcut : non-successive mask transfer and interpolation for video cutout. *ACM Trans. Graph.*, 34:195 :1–195 :10.

[Fang *et al.*, 2017] Fang, H., Xie, S., Tai, Y.-W. et Lu, C. (2017). Rmpe : Regional multi-person pose estimation. *IEEE International Conference on Computer Vision (ICCV)*, pages 2353–2362.

[Felzenszwalb et Huttenlocher, 2005] Felzenszwalb, P. et Huttenlocher, D. (2005). Pictorial structures for object recognition. *International Journal of Computer Vision*, 61:55–79.

[Felzenszwalb *et al.*, 2010] Felzenszwalb, P. F., Girshick, R. B., McAllester, D. et Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645.

[Feng, 2022] Feng, Y. (2022). *Deep Learning Approach for Semantic Segmentation of Pathological Images*. Thèse de doctorat, INSA Centre Val de Loire.

[Fieraru *et al.*, 2018] Fieraru, M., Khoreva, A., Pishchulin, L. et Schiele, B. (2018). Learning to refine human pose estimation. *In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

[Fukushima, 2004] Fukushima, K. (2004). Neocognitron : A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202.

[Girdhar *et al.*, 2018] Girdhar, R., Gkioxari, G., Torresani, L., Paluri, M. et Tran, D. (2018). Detect-and-Track : Efficient Pose Estimation in Videos. *In Computer Vision and Pattern Recognition (CVPR)*.

[Girshick, 2015] Girshick, R. (2015). Fast r-cnn. *In International Conference on Computer Vision (ICCV)*.

[Girshick *et al.*, 2018] Girshick, R., Radosavovic, I., Gkioxari, G., Dollár, P. et He, K. (2018). Detectron. https://github.com/facebookresearch/detectron.

[Girshick *et al.*, 2014] Girshick, R. B., Donahue, J., Darrell, T. et Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587.

[Grundmann *et al.*, 2010] Grundmann, M., Kwatra, V., Han, M. et Essa, I. (2010). Efficient hierarchical graph-based video segmentation. *In IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2141–2148.

[He *et al.*, 2017] He, K., Gkioxari, G., Dollar, P. et Girshick, R. (2017). Mask r-CNN. *In 2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988. IEEE.

[He *et al.*, 2016] He, K., Zhang, X., Ren, S. et Sun, J. (2016). Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

[Hinton, 2012] Hinton, G. (2012). Lecture 6.5 - rmsprop, coursera : Neural networks for machine learning.

[Hirose *et al.*, 2018] Hirose, N., Sadeghian, A., Vázquez, M., Goebel, P. et Savarese, S. (2018). Gonet : A semi-supervised deep learning approach for traversability estimation. *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 3044–3051.

[Hogg, 1983] Hogg, D. C. (1983). Model-based vision : a program to see a walking person. *Image and Vision Computing*, 1:5–20.

[Hu *et al.*, 2017] Hu, Y.-T., Huang, J.-B. et Schwing, A. G. (2017). Maskrnn : Instance level video object segmentation. *In Advances in Neural Information Processing Systems (NeurIPS)*.

[Hu *et al.*, 2018] Hu, Y.-T., Huang, J.-B. et Schwing, A. G. (2018). Videomatch : Matching based video object segmentation. *ArXiv*, abs/1809.01123.

[Hubel et Wiesel, 1968] Hubel, D. et Wiesel, T. (1968). Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195:215–243.

[Insafutdinov *et al.*, 2017] Insafutdinov, E., Andriluka, M., Pishchulin, L., Tang, S., Levinkov, E., Andres, B. et Schiele, B. (2017). Arttrack : Articulated multi-person tracking in the wild. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Insafutdinov *et al.*, 2016] Insafutdinov, E., Pishchulin, L., Andres, B., Andriluka, M. et Schieke, B. (2016). Deepercut : A deeper, stronger, and faster multi-person pose estimation model. *In European Conference on Computer Vision (ECCV)*.

[Ioffe et Forsyth, 2001] Ioffe, S. et Forsyth, D. (2001). Human tracking with mixtures of trees. *In Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, volume 1, pages 690–695 vol.1.

[Iqbal *et al.*, 2017] Iqbal, U., Milan, A. et Gall, J. (2017). Posetrack : Joint multi-person pose estimation and tracking. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4654–4663.

[Jampani *et al.*, 2017] Jampani, V., Gadde, R. et Gehler, P. V. (2017). Video propagation networks. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Jin *et al.*, 2019] Jin, S., Liu, W., Ouyang, W. et Qian, C. (2019). Multi-person articulated tracking with spatial and temporal embeddings. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5657–5666.

[Kannala et Brandt, 2006] Kannala, J. et Brandt, S. (2006). A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *IEEE transactions on pattern analysis and machine intelligence*, 28:1335–40.

[Khoreva *et al.*, 2019] Khoreva, A., Benenson, R., Ilg, E., Brox, T. et Schiele, B. (2019). Lucid data dreaming for video object segmentation. *International Journal of Computer Vision*, pages 1–23.

[Kingma et Ba, 2015] Kingma, D. P. et Ba, J. (2015). Adam : A method for stochastic optimization. *In International Conference on Learning Representations (ICLR)*.

[Kreiss *et al.*, 2019] KREISS, S., BERTONI, L. et ALAHI, A. (2019). Pifpaf : Composite fields for human pose estimation. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Krizhevsky, 2009] KRIZHEVSKY, A. (2009). Learning multiple layers of features from tiny images. *Master's thesis, Computer Science Department, University of Toronto*.

[Krizhevsky *et al.*, 2012] KRIZHEVSKY, A., SUTSKEVER, I. et HINTON, G. E. (2012). Imagenet classification with deep convolutional neural networks. *In Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, page 1097–1105.

[Kumar *et al.*, 2016] KUMAR, A., IRSOY, O., ONDRUSKA, P., IYYER, M., BRADBURY, J., GULRAJANI, I., ZHONG, V., PAULUS, R. et SOCHER, R. (2016). Ask me anything : Dynamic memory networks for natural language processing. *In Proceedings of The 33rd International Conference on Machine Learning*, volume 48, pages 1378–1387. PMLR.

[LeCun *et al.*, 1989] LECUN, Y., BOSER, B., DENKER, J. S., HENDERSON, D., HOWARD, R. E., HUBBARD, W. et JACKEL, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551.

[Lecun *et al.*, 1998] LECUN, Y., BOTTOU, L., BENGIO, Y. et HAFFNER, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[LeCun *et al.*, 2004] LECUN, Y., HUANG, F. J. et BOTTOU, L. (2004). Learning methods for generic object recognition with invariance to pose and lighting. *In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages II–104 Vol.2.

[Li et Loy, 2018] LI, X. et LOY, C. C. (2018). Video object segmentation with joint re-identification and attention-aware mask propagation. *In European Conference on Computer Vision (ECCV)*.

[Lin *et al.*, 2017a] LIN, T.-Y., DOLLÁR, P., GIRSHICK, R., HE, K., HARIHARAN, B. et BELONGIE, S. (2017a). Feature pyramid networks for object detection.

[Lin *et al.*, 2017b] LIN, T.-Y., GOYAL, P., GIRSHICK, R., HE, K. et DOLLAR, P. (2017b). Focal loss for dense object detection. *In Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

[Lin *et al.*, 2014] LIN, T.-Y., MAIRE, M., BELONGIE, S., BOURDEV, L., GIRSHICK, R., HAYS, J., PERONA, P., RAMANAN, D., ZITNICK, C. L. et DOLLÁR, P. (2014). Microsoft COCO : Common objects in context. *European Conference on Computer Vision (ECCV)*.

[Lin *et al.*, 2011] Lin, Y., Lv, F., Zhu, S., Yang, M., Cour, T., Yu, K., Cao, L. et Huang, T. (2011). Large-scale image classification : Fast feature extraction and svm training. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1689–1696.

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

[Maji, 2011] Maji, S. (2011). Large scale image annotations on amazon mechanical turk. Rapport technique UCB/EECS-2011-79, EECS Department, University of California, Berkeley.

[Mamoru Saito *et al.*, 2010] Mamoru Saito, Katsuhisa Kitaguchi, Gun Kimura et Masafumi Hashimoto (2010). Human detection from fish-eye image by bayesian combination of probabilistic appearance models. *In IEEE International Conference on Systems, Man and Cybernetics*, pages 243–248.

[Maninis *et al.*, 2018] Maninis, K.-K., Caelles, S., Chen, Y., Pont-Tuset, J., Leal-Taixé, L., Cremers, D. et Van Gool, L. (2018). Video object segmentation without temporal information. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*.

[Manovich et Sekachev, 2019] Manovich, N. et Sekachev, B. (2019). Powerful and efficient computer vision annotation tool (cvat). https://github.com/opencv/cvat.

[Marcos *et al.*, 2016] Marcos, D., Volpi, M. et Tuia, D. (2016). Learning rotation invariant convolutional filters for texture classification. *In International Conference on Pattern Recognition (ICPR)*, pages 2012–2017. IEEE.

[Miller *et al.*, 2016] Miller, A., Fisch, A., Dodge, J., Karimi, A.-H., Bordes, A. et Weston, J. (2016). Key-value memory networks for directly reading documents. *In Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409.

[Miller, 1995] Miller, G. A. (1995). Wordnet : A lexical database for english. *Commun. ACM*, 38(11):39–41.

[Minsky et Papert, 1987] Minsky, M. et Papert, S. (1987). *Perceptrons : an introduction to computational geometry*. The MIT Press.

[Mori et Malik, 2002] Mori, G. et Malik, J. (2002). Estimating human body configurations using shape context matching. *In European Conference on Computer Vision (ECCV)*.

[Morris et Rehg, 1998] Morris, D. et Rehg, J. (1998). Singularity analysis for articulated object tracking. *In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 289–296.

[Märki *et al.*, 2016] Märki, N., Perazzi, F., Wang, O. et Sorkine-Hornung, A. (2016). Bilateral space video segmentation. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 743–751.

[Newell *et al.*, 2016] Newell, A., Yang, K. et Deng, J. (2016). Stacked hourglass networks for human pose estimation. *In European Conference on Computer Vision (ECCV)*, pages 483–499.

[Ning et Huang, 2020] Ning, G. et Huang, H. (2020). Lighttrack : A generic framework for online top-down human pose tracking. *Proceedings of CVPRW 2020 on Towards Human-Centric Image/Video Synthesis and the 4th Look Into Person (LIP) Challenge*.

[Nwankpa *et al.*, 2018] Nwankpa, C., Ijomah, W. L., Gachagan, A. et Marshall, S. (2018). Activation functions : Comparison of trends in practice and research for deep learning. *ArXiv*, abs/1811.03378.

[Odemakinde,] Odemakinde, E. https://viso.ai/deep-learning/mask-r-cnn/.

[Oh *et al.*, 2018] Oh, S. W., Lee, J.-Y., Sunkavalli, K. et Kim, S. J. (2018). Fast video object segmentation by reference-guided mask propagation. *In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7376–7385.

[Oh *et al.*, 2019] Oh, S. W., Lee, J.-Y., Xu, N. et Kim, S. J. (2019). Video object segmentation using spacetime memory networks. *International Conference on Computer Vision (ICCV)*.

[O'Rourke et Badler, 1980] O'Rourke, J. et Badler, N. (1980). Model-based image analysis of human motion using constraint propagation. *PAMI*, 2:522–536.

[Perazzi *et al.*, 2017] Perazzi, F., Khoreva, A., Benenson, R., Schiele, B. et Sorkine-Hornung, A. (2017). Learning video object segmentation from static images. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3491–3500.

[Pishchulin *et al.*, 2013] Pishchulin, L., Andriluka, M., Gehler, P. et Schiele, B. (2013). Poselet conditioned pictorial structures. *In 2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 588–595.

[Pont-Tuset *et al.*, 2017] Pont-Tuset, J., Perazzi, F., Caelles, S., Arbeláez, P., Sorkine-Hornung, A. et Van Gool, L. (2017). The 2017 davis challenge on video object segmentation. *arXiv :1704.00675*.

[Raaj *et al.*, 2019] Raaj, Y., Idrees, H., Hidalgo, G. et Sheikh, Y. (2019). Efficient online multi-person 2d pose tracking with recurrent spatio-temporal affinity fields. *arXiv :1811.11975*.

[Ramakanth et Babu, 2014] Ramakanth, S. A. et Babu, R. V. (2014). Seamseg : Video object segmentation using patch seams. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 376–383.

[Ren *et al.*, 2015] Ren, S., He, K., Girshick, R. et Sun, J. (2015). Faster r-cnn : Towards real-time object detection with region proposal networks. *In Advances in Neural Information Processing Systems (NeurIPS)*, volume 28, pages 91–99.

[Ren *et al.*, 2005] Ren, X., Berg, A. et Malik, J. (2005). Recovering human body configurations using pairwise constraints between parts. *In IEEE International Conference on Computer Vision (ICCV)*, volume 1, pages 824–831 Vol. 1.

[Rosenblatt, 1958] Rosenblatt, F. (1958). The perceptron : A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408.

[Rumelhart *et al.*, 1986] Rumelhart, D. E., Hinton, G. E. et Williams, R. J. (1986). *Learning Internal Representations by Error Propagation*, page 318–362. MIT Press, Cambridge, MA, USA.

[Saez *et al.*, 2019] Saez, A., Bergasa, L., Lopez-Guillen, E., Romera, E., Tradacete, M., Gomez-Huélamo, C. et del Egido, J. (2019). Real-time semantic segmentation for fisheye urban driving images based on ERFNet. *Sensors*, 19(3):503.

[Samuel, 1959] Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229.

[Sanz-Ablanedo *et al.*, 2010] Sanz-Ablanedo, E., Rodriguez-Pérez, J. R., Armesto, J. et Taboada, M. F. A. (2010). Geometric stability and lens decentering in compact digital cameras. *Sensors*, 10(3):1553–1572.

[Sekkat *et al.*, 2020] Sekkat, A. R., Dupuis, Y., Vasseur, P. et Honeine, P. (2020). The omniscape dataset. *In International Conference on Robotics and Automation (ICRA)*, pages 1603–1608.

[Simonyan et Zisserman, 2015] Simonyan, K. et Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *In International Conference on Learning Representations (ICLR)*.

[Snower *et al.*, 2020] Snower, M., Kadav, A., Lai, F. et Graf, H. P. (2020). 15 keypoints is all you need. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Sorokin et Forsyth, 2008] Sorokin, A. et Forsyth, D. (2008). Utility data annotation with amazon mechanical turk. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 1–8.

[Sukhbaatar *et al.*, 2015] Sukhbaatar, S., Szlam, A., Weston, J. et Fergus, R. (2015). End-to-end memory networks. *In Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, page 2440–2448.

[Sun *et al.*, 2019] Sun, K., Xiao, B., Liu, D. et Wang, J. (2019). Deep high-resolution representation learning for human pose estimation. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Toshev et Szegedy, 2014] Toshev, A. et Szegedy, C. (2014). Deeppose : Human pose estimation via deep neural networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1653–1660.

[Uijlings *et al.*, 2013] Uijlings, J. R. R., van de Sande, K. E. A., Gevers, T. et Smeulders, A. W. M. (2013). Selective search for object recognition. *International Journal of Computer Vision*, 104:154–171.

[Velastin et Gómez-Lira, 2017] Velastin, S. A. et Gómez-Lira, D. A. (2017). People detection and pose classification inside a moving train using computer vision. *In International Visual Informatics Conference*, pages 319–330. Springer.

[Wang *et al.*, 2019] Wang, T., Hsieh, Y., Wong, F. et Chen, Y. (2019). Mask-rcnn based people detection using a top-view fisheye camera. *In International Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, pages 1–4.

[Wei *et al.*, 2016] Wei, S.-E., Ramakrishna, V., Kanade, T. et Sheikh, Y. (2016). Convolutional pose machines. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Wikipedia, 2022] Wikipedia (2022). Support-vector machine — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Support-vector%20machine&oldid=1107938510.

[Xiao *et al.*, 2018] Xiao, B., Wu, H. et Wei, Y. (2018). Simple baselines for human pose estimation and tracking. *In European Conference on Computer Vision (ECCV)*.

[Xiu *et al.*, 2018] Xiu, Y., Li, J., Wang, H., Fang, Y. et Lu, C. (2018). Pose Flow : Efficient online pose tracking. *In British Machine Vision Conference (BMVC)*.

[Xu *et al.*, 2018] Xu, N., Yang, L., Fan, Y., Yue, D., Liang, Y., Yang, J. et Huang, T. (2018). Youtube-vos : A large-scale video object segmentation benchmark. *arXiv :1809.03327*.

[Yamamoto et Koshikawa, 1991] Yamamoto, M. et Koshikawa, K. (1991). Human motion analysis based on a robot arm model. *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 664–665.

[Yang *et al.*, 2018] YANG, L., WANG, Y., XIONG, X., YANG, J. et KATSAGGELOS, A. K. (2018). Efficient video object segmentation via network modulation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6499–6507.

[Yang *et al.*, 2017] YANG, W., LI, S., OUYANG, W., LI, H. et WANG, X. (2017). Learning feature pyramids for human pose estimation. *IEEE International Conference on Computer Vision (ICCV)*, pages 1290–1299.

[Yang et Ramanan, 2013] YANG, Y. et RAMANAN, D. (2013). Articulated human detection with flexible mixtures of parts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2878–2890.

[Yogamani *et al.*, 2019] YOGAMANI, S., HUGHES, C., HORGAN, J., SISTU, G., VARLEY, P., O'DEA, D., URICÁR, M., MILZ, S., SIMON, M., AMENDE, K. *et al.* (2019). Woodscape : A multi-task, multi-camera fisheye dataset for autonomous driving. *arXiv preprint arXiv :1905.01489*.

[Yoon *et al.*, 2017] YOON, J. S., RAMEAU, F., KIM, J., LEE, S., SHIN, S. et KWEON, I.-S. (2017). Pixel-level matching for video object segmentation using convolutional neural networks. *IEEE International Conference on Computer Vision (ICCV)*, pages 2186–2195.

[Zhang *et al.*, 2016] ZHANG, Z., REBECQ, H., FORSTER, C. et SCARAMUZZA, D. (2016). Benefit of large field-of-view cameras for visual odometry. *In ICRA*.

[Zhou et Chellappa, 1988] ZHOU et CHELLAPPA (1988). Computation of optical flow using a neural network. *In IEEE International Conference on Neural Networks*, pages 71–78 vol.2.

Table des figures

Liste des tableaux

Liste des publications de l'auteur

Conférences internationales

- [Dufour *et al.*, 2020] Rémi Dufour, Cyril Meurie, Clément Strauss, Olivier Lezoray. Instance segmentation in fisheye images. IPTA 2020, International Conference on Image Processing Theory, Tools and Applications, Nov 2020, Paris, France.
- [Dufour *et al.*, 2022] Rémi Dufour, Cyril Meurie, Olivier Lézoray, Ankur Mahtani. Space-Time Memory networks for multi-person skeleton body part detection. 3rd International Conference on Pattern Recognition and Artificial Intelligence ICPRAI 2022, Vol. LNCS 13364, pp. 78—90, Jun 2022, Paris, France.

Conférences nationales

- [Dufour *et al.*, 2021] Rémi Dufour, Cyril Meurie, Olivier Lézoray, Ankur Mahtani. Segmentation d'instance dans des images fisheye. ORASIS 2021, Centre National de la Recherche Scientifique (CNRS), Sep 2021, Saint Ferréol, France.

Glossaire

AdaGrad Adaptive Gradient : un algorithme d'optimisation efficace pour traiter des gradients épars.. 29, 149

Adam Adaptive Moment estimation : un algorithme d'optimisation stochastique populaire qui combine les avantages de AdaGrad et RMSProp.. 29, 33

CNN Convolutional Neural Networks : les réseaux de neurones convolutifs.. 24, 26, 33, 36, 38, 39, 40, 50, 62, 63, 64, 67, 71

DNN Deep Neural Network : un réseau de neurones profond.. 62, 63, 77

FE Fisheye Effect : des images présentant des distorsions caractéristiques des images capturées à l'aide d'un objectif fisheye.. 14, 15, 36, 38, 39, 42, 43, 45, 47, 48, 49, 50, 51, 52, 57, 120, 121, 122, 123, 140, 145

FPN Feature Pyramid Network : une méthode qui profite de la hiérarchie en pyramide d'un réseau convolutif et des connexions latérales pour construire des cartes de caractéristiques en pyramide, capable de traiter plusieurs échelles.. 40

GAN Generative Adversarial Network : un réseau de neurone entraîné contre un réseau de neurone adverse pour la génération d'image.. 38

HE Human Embedding : une extension de KE qui capture des caractéristiques globale d'apparence au niveau des humains.. 67

HOG Histogram of Oriented Gradients : une caractéristique qui fonctionne en calculant des histogrammes locaux du gradient dans une grille dense.. 39

IEF Iterative Error Feedback : un processus dans lequel le modèle corrige progressivement ses erreurs.. 62

KE Keypoint Embedding : un embedding produit par un modèle de détection de points clés du corps humain qui permet de grouper les points clés pour former des squelettes complets.. 67, 69, 149

ML Machine Learning : le champ d'étude de l'intelligence artificielle qui vise à donner aux machines la capacité d'apprendre à partir de données.. 17

NLP Natural Language Processing : le domaine de recherche qui concerne le traitement automatisé du language naturel.. 67, 72

NMS Non-Maximum Suppression : une technique qui consiste à supprimer, parmi des détections qui se superposent, celles qui ont un score de confiance inférieur à une autre.. 63

PAF Part Affinity Fields : des champs vectoriels indiquant à la fois la confiance et la direction d'une arête de squelette du corps humain.. 63, 64, 65, 67, 68, 69, 76, 141

PIF Part Intensity Fields : qui consiste en une fusion d'une carte de confiance de points clés avec un composant vectoriel qui pointe vers la partie du corps la plus proche, et avec un autre composant scalaire qui indique la taille du point clé.. 65

PRM Pyramid Residual Module : un module qui apprend des filtres convolutifs pour différentes échelles des caractéristiques d'entrée.. 67

RMPE Regional Multi-Person Pose Estimation : un algorithme top-down d'estimation de pose capable de traiter plusieurs personnes.. 66

RMSProp Root Mean Square Propagation : un algorithme d'optimisation efficace pour traiter des objectifs non-stationnaires.. 29, 149

RPN Region Proposal Network : un composant de Faster R-CNN qui génère des propositions de boîtes englobantes entourant des objets détectés dans l'image.. 40, 71

SGD Stochastic Gradient Descent : un algorithme pour optimiser une fonction de loss qui utilise une estimation du gradient obtenue grâce à un échantillonnage aléatoire des données.. 27, 29, 33

SIE Spatial Instance Embedding : une carte 2D d'embedding où chaque pixel est encodé avec la prédiction du centre de la personne, pour faciliter le groupement de points clés en squelettes complets.. 67, 69, 151

SIFT Scale-Invariant Feature Transform : un algorithme pour détecter et identifier les éléments similaires entre deux images.. 39

SLAM Simultaneous Localization And Mapping : la tâche pour un robot de simultanément se localiser et cartographier son environnement.. 38

SPM Scaled Prismatic Model : un modèle de recalage 2D.. 61

STAF Spatio-Temporal Affinity Fields : un algorithme bottom-up de suivi de pose.. 67, 68, 69, 141

STM Space-Time Memory : une architecture de réseaux de neurones profonds conçue pour la tâche de VOS.. 14, 60, 72, 74, 75, 76, 77, 79, 80, 83, 84, 85, 86, 88, 95, 122, 141, 142, 145

SVM Support-Vector Machine : une technique d'apprentissage supervisé pour résoudre des problèmes de classification et de régression.. 18, 19, 20, 32, 139

TAF Temporal Affinity Fields : des champs vectoriels indiquant à la fois la confiance et la direction d'une arête de squelette du corps humain, en prenant en compte la dimension temporelle.. 67, 68, 69, 141

TFF Temporal Flow Fields : une représentation du mouvement de chaque point clé du corps utilisant un ensemble de vecteurs 2D.. 69

TIE Temporal Instance Embedding : une extension de SIE qui concatène des caractéristiques bas niveau, des cartes de confiance de points clés de squelette et des SIE de deux frames consécutives.. 67

VOS Video Object Segmentation : la tâche qui consiste à segmenter des objets dans des vidéos.. 14, 59, 60, 69, 70, 71, 72, 74, 141, 151

VT Vérité Terrain : correspond à une référence réalisée à la main par un ou des experts.. 77, 82, 83

Résumé

Les projets de trains autonomes se multiplient à travers le monde. En France un consortium dirigé par l'IRT Railenium a pour objectif de construire un prototype de train autonome atteignant GoA4 (Grade of Automation 4) et qui serait capable de circuler sans conducteur et sans personnel humain à bord. En l'absence de personnel, les besoins de services et de sécurité des usagers doivent être pris en charge par des systèmes automatisés. De tels systèmes doivent disposer d'informations variées et détaillées, en particulier sur l'état et les actions des usagers présents à bord. Les algorithmes de vision par ordinateur, en particulier ceux basés sur l'apprentissage automatique par réseaux de neurones profonds, aussi appelés Deep Learning, ont récemment atteint des niveaux de performances convenables pour analyser des flux vidéos de caméras de surveillance. Plusieurs défis spécifiques au contexte du train autonome doivent cependant être relevés. Certaines caméras chargées d'assurer la sécurité des passagers à l'intérieur du train seront de type grand angle ou fisheye. Ces dernières produisent des images présentant des distorsions en barillet importantes, qui ne sont pas présentes dans les principales bases de données d'entraînement de la littérature, et qui permettent l'apprentissage des réseaux de neurones convolutifs modernes. Dans la littérature, une méthode a été développée pour entraîner des algorithmes de segmentation sémantique sur des images fisheye artificielles. Dans cette thèse, nous appliquons pour la première fois, cette méthode à la tâche de segmentation d'instance, et nous étudions d'une part ses performances sur deux nouvelles bases d'images réelles présentant des distorsions en barillet, mais aussi l'effet de l'initialisation et de certains paramètres. De plus, les nouveaux algorithmes de suivi de pose ont maintenant atteint une certaine maturité. Cependant, ils sont généralement top-down, et ne disposent pas d'une mémoire à long terme. Nous proposons donc dans un second temps une nouvelle méthode de détection de points clés de squelettes de personnes dans des vidéos, en adaptant un algorithme récent de Video Object Segmentation (VOS), qui dispose d'une mémoire à long terme. Ces propositions d'algorithmes ont été évaluées et validées sur des données acquises dans un modèle de train Regio2N spécifique au contexte du futur train autonome.

Indexation RAMEAU : Vision par ordinateur, Apprentissage profond, Traitement d'images - Techniques numériques, Vidéosurveillance, Reconnaissance de l'activité humaine (informatique)

Abstract

Autonomous train projects are multiplying around the world. In France a consortium directed by IRT Railenium has the goal to build a train prototype that achieves GoA4 (Grade of Automation 4) that would be able to navigate without a driver and without on-board staff. In the absence of staff, the needs of services and security of the passengers must be taken care of by automatic systems. Such systems must have varied and detailed information, in particular on the state and actions of the passengers on board. Computer vision algorithms, in particular those based on machine larning by deep neural networks, also called "Deep Learning" algorithms, have recently achieved a level of performance adequate to analyse video streams from surveillance cameras. However, many challenges specific to the context of autonommous trains need to be adressed. Some cameras tasked with ensuring the security of passengers inside the train will be of the wide-angle or Fisheye type. Those cameras produce images that contain important barrel distortions, that are not present in the main datasets from the literature that permit the training of modern convolutional neural networks. In the literature, a method was developed to train semantic segmentation algorithm on artificial fisheye images. In this thesis, we apply for the first time this method for the task of instance segmentation, and we study, on the one hand, its performance on two new annotated real images datasets with barrel distortions, but also the effect of initialization and certain parameters. Moreover, recent pose tracking algorithms have now achieved some degree of maturity. However, they are generally top-down, and do not make use of a long term memory. We propose a new method for person skeleton key-points detection in video, that adapts a recent Video Object Segmentation (VOS) algorithm, that makes use of a long term memory. These algorithms propositions were evaluated and validated on data acquired in a train model Regio2N specific to the context of the future autonomous train.

RAMEAU Index : Computer vision, Deep learning, Image processing - Numerical techniques, Videosurveillance, Human action recognition (computer science)