UV6.1 - Projet de fin d'études

Deep learning based risk assessment algorithms near automatic doors

Auteur :
M. Olivier Laurendin

Encadrants :
M. Luc Jaulin
M. Sebastien Lefèbvre
M. Clément Strauss
M. Sébastien Ambéllouis

mars 2019 - septembre 2019
20 août 2019

Remerciements

Mes remerciements s'adressent tout d'abord à M. Sébastien Lefèbvre et M. Clément Strauss pour leur encadrement et leur présence au quotidien tout au long du stage. Je tiens aussi à remercier M. Sébastien Ambéllouis pour son soutien et ses orientations dans les choix technologiques effectués.

Un grand merci également à M. Luc Jaulin, M. Fabrice Le Bars, M. Benoît Zerr, M. Gilles Le Chenadec et M. Laurent Hardouin pour l'enseignement qu'ils m'ont dispensé pendant mes années à l'ENSTA Bretagne et à l'université d'Angers qui s'est avéré être d'une utilité majeure pendant ce stage.

Je souhaite également remercier mes collègues de travail et les stagiaires de chez Railenium, pour leur aide et les nombreuses discussions que nous avons partagé. Notamment à M^{lle} Hiba Bouchama, M. Mohamed Amine Hadded, M. Nabil Hentati, M. Quentin Gadmer, M. Ilyes Hamitouche et M. Yasser Hamidullah.

J'adresse aussi un remerciement tout particulier au pôle administratif de Railenium, notamment M^{me} Lucille Guerroumi et à M. Rudy Dahyot pour la patience et la bienveillance dont ils ont fait preuve à mon égard.

Enfin, j'aimerais remercier M. Abderraouf Boussif pour ses explications précieuses sur les niveaux de danger autour des portes automatiques et son aide pour le défrichage du sujet du stage.

Table des matières

Résumé

Avec le nombre de voyageurs sur le réseau ferré français en constante augmentation atteignant plus de 1,2 milliards d'usagers pour 91,5 milliards de kilomètres parcourus en 2018 pour la SNCF seule [29], les pressions sur les infrastructures ferroviaires françaises amènent à des réflexions sur une modernisation de celles-ci. Parmi les solutions possibles, l'augmentation du flux de passagers sur des lignes ferroviaires déjà existantes nécessite une automatisation partielle, sinon totale, de l'opération des trains. Permettant de réduire les temps d'arrêt en gare et les éventuelles erreurs humaines pouvant mener à une paralysie du réseau. Le stage que j'ai effectué chez Railenium s'inscrit dans un projet visant à produire un train autonome circulant sur les rames grande vitesse similaire aux trains automatiques utilisés depuis plusieurs années dans le métro parisien. La transition de ce système de rames de métro automatiques à des rames grande vitesse autonome possède des contraintes nouvelles dont celle développée ici, à savoir l'identification de situations de mise en danger d'usagers dans le voisinage direct des portes automatiques du train. Ce résumant dans le cas présent à l'identification et au suivi de trajectoires de piétons à l'approche de portes automatiques dans un environnement variable. Le présent rapport présente une première implémentation d'un algorithme de suivi de piétons qui sera à terme employé pour traiter les flux vidéos de deux caméras fisheye placées à l'intérieur d'un wagon de train.

Mots clés : Suivi Multi Objets, Suivi de Piétons, Identification de Dangers, Train Autonome, Apprentissage Profond.

Abstract

With the ever-increasing number of passengers on the French rail network reaching more than 1.2 billion users for 91.5 billion kilometres travelled in 2018 for SNCF alone [29], the strain on French rail infrastructure is leading to considerations on its modernization. Among the possible solutions, increasing passenger flow on existing rail lines requires partial, if not total, automation of train operation. In order to reduce station downtime and possible human errors that could lead to network paralysis. The internship I did at Railenium is part of a project to produce an autonomous high speed train similar to the automatic trains used for several years in the Paris subway system. The transition from this system of automatic subway trains to autonomous high speed trains involves new constraints, including the one developed here, namely danger assessment in the direct vicinity of the train's automatic doors for users safety. In our case study, it will be narrowed down to identifying and monitoring pedestrian trajectories when approaching automatic doors in a variable environment. This report introduces a first implementation of a multi object tracking algorithm that will eventually be used to process video streams from two fisheye cameras placed in a train car.

Key words : Multi Object Tracking, Pedestrians Tracking, Danger Assessment, Autonomous Train, Deep Learning.

Introduction

Présentation Railenium et IFSTTAR

J'ai réalisé mon stage au sein de l'institut de recherche technologique Railenium basé à Valenciennes dans les Hauts-de-France . L'IRT Railenium est un centre de recherche appliquée qui conduit des projets de recherche et développement dans les domaines des systèmes, de la maintenance, de la conception de l'infrastructure, du matériel roulant et de la gestion de l'énergie dans le domaine ferroviaire.

Créé en octobre 2012 sous la forme d'une Fondation de Coopération Scientifique, Railenium est l'un des 8 Instituts de Recherche Technologique (IRT), constitué dans le cadre du Programme des Investissements d'Avenir (PIA). Il réunit aujourd'hui 28 membres dont :

- Le pôle de compétitivité i-Trans, dont Railenium est né,
- Des exploitants ferroviaires tels que SNCF ou Eurotunnel,
- Des industriels, constructeurs et bureaux d'ingénierie tels que Alstom ou Thalès,
- Des centres publics de recherche et de formation tels que l'IFSTTAR, les Universités de Valenciennes et du Hainaut-Cambrésis, Centrale Lille ou encore l'institut Mines-Télécom[49].

Parmi les projets gérés par l'IRT Railenium, j'ai travaillé sur le projet Train Autonome qui vise à la mise en circulation de trains de voyageurs semi-autonomes d'ici 2020 et des trains entièrement autonomes d'ici 2023. L'IRT Railenium se place pour cela aux côtés de la SNCF au sein d'un consortium d'industriels composé des entreprises Bombardier, Bosch, Spirops et Thalès pour le développement d'un prototype de train autonome dédié au transport de voyageurs.

Les objectifs affichés de cette innovation sont divers :

- Une plus grande fluidité et régularité des trains grâce à une circulation harmonisée et une vitesse optimisée,
- Une plus grande capacité de transport des infrastructures ferroviaires grâce à une augmentation du trafic ferroviaire rendu possible par l'automatisation,
- Un impact écologique plus faible grâce à une diminution de la consommation d'énergie,
- Un plus grand confort pour les usagers en leur proposant des services autonomes et

personnalisés à bord du train[56].

Ce stage s'est effectué en collaboration avec L'Institut Français des Sciences et Technologies des Transports, de l'Aménagement et des Réseaux (IFSTTAR). l'IFSTTAR est un centre de recherche dans les transports, les infrastructures, les risques naturels et la ville pour améliorer les conditions de vie des citadins et plus largement favoriser un développement durable des sociétés basé à Villeneuve d'Ascq.

Durant mon stage, j'ai été affilié au lot 13 du projet train autonome intitulé "Détection d'Obstacles au niveau des Portes" qui vise à qualifier l'ensemble des situations de danger aux alentours des portes automatiques du futur train autonome afin d'assurer la sécurité des usagers lors de l'ouverture ou de la fermeture des portes du train.

Définition niveau d'autonomie d'un véhicule

Afin de comprendre les enjeux autour du train autonome, il nous faut d'abord définir le concept de train autonome en s'intéressant aux différents niveaux d'autonomie, ou Grade of Automation (GoA) existants dans le domaine ferroviaire. Le tableau 2 résume les différents niveaux d'autonomie possibles du train autonome. Le niveau GoA0 correspondant à une simple assistance signalétique au conducteur et d'une gestion de l'aiguillage par le système autonome.
Au niveau GoA1, le train est en mesure de contrôler sa vitesse en déplacement nominal, d'une manière similaire à un régulateur de vitesse de voiture.
Le niveau GoA2 correspond au niveau d'autonomie permettant la conduite "sans les mains". Le déplacement du train est intégralement géré de manière autonome. Le conducteur reste en cabine pour pouvoir gérer les autres tâches telles que l'ouverture des portes ou la surveillance de l'environnement.
Au niveau GoA3, le train est équipé d'un système autonome de gestion des collisions et de détection des obstacles. Il reste à bord du train un chef de bord qui est là pour reprendre la main en cas de situation d'urgence.
Le niveau GoA4 est le niveau d'autonomie maximale où aucune personne de l'entreprise ferroviaire n'est requise à son bord.[56]

Chaque montée en autonomie requiert une diminution des taux d'erreur des systèmes embarqués. Railenium vise le développement de trains de niveau GoA2 d'ici 2023 et jusqu'au niveau GoA4 d'ici 2025 afin de permettre à ses exploitants ferroviaires partenaires comme la SNCF de choisir le niveau d'autonomie qu'ils souhaitent utiliser pour chaque trame[56].

En réalité, des technologies proches au train autonome existent déjà. On peut notamment citer les 55 lignes de métro automatisées présentes dans 37 villes du monde en Juillet

Niveau	Description	Présence de personnel
GoA0	Conduite à vue, l'aiguillage est géré par le système	Cabine + wagons
GoA1	Le système garantit la vitesse sécuritaire du train	Cabine + wagons
GoA2	Accélération et freinage autonomes	(Cabine) + wagons
GoA3	Prévention autonome de collision avec des obstacles	Wagons
GoA4	Autonomie du démarrage du train, de la supervision de l'échange, de la sécurité et des services aux voyageurs et de la gestion des situations d'urgence	Sur les quais

FIGURE 2 – Tableau récapitulatif des différents niveaux d'autonomie possibles du train autonome.

2016 selon l'Union Internationale des Transports Publics [20]. Le projet du train autonome consiste donc en une adaptation des technologies existantes du métro automatique sur des lignes grande vitesse. Cette adaptation amène cependant son lot de difficultés et de verrous technologiques nouveaux. En effet, contrairement au métro sous-terrain, les lignes grande vitesse évoluent dans un environnement incertain et ouvert. De nouveaux besoins, comme le monitoring de l'environnement ou la détection d'obstacles apparaissent alors comme des priorités sécuritaires. La détection d'obstacles à la fermeture des portes automatiques du train est une difficulté qui n'a été que partiellement résolue sur les métros automatiques, et reste encore aujourd'hui une source de risque pour les voyageurs. L'intervention humaine étant encore requise pour sortir un voyageur pris dans les portes par exemple. La transition vers un train totalement autonome nécessite alors le développement de nouveaux algorithmes garantissant le bon déroulement de la fermeture des portes automatiques avant le départ du train.

Les objectifs du stage sont alors multiples :

— Définir les besoins sécuritaires autour de la fermeture des portes automatiques en listant les dangers potentiels aux alentours des portes lors de leur fermeture. On classera alors ces besoins en fonction de leur fréquence et de leur niveau de dangerosité afin de prioriser les algorithmes à développer,
— Effectuer un état de l'art des algorithmes permettant de répondre au besoin le plus important selon la liste précédente,
— Implémenter une première chaîne de traitement, évaluer son efficacité sur des exemples pertinents et décrire les pistes d'amélioration possibles.

Les pistes d'amélioration citées ainsi que les algorithmes restant à développer feront l'objet d'une thèse débutant en novembre 2019 en collaboration entre l'IRT Railenium et le laboratoire IFSTTAR pour une période de trois ans.

Qualification du besoin

A/ Définition du besoin

La première partie du stage consiste en la définition des tâches que les algorithmes développés dans le cadre de la sécurité des passagers aux alentours des portes automatiques doivent être capables d'effectuer. Cela passe dans un premier temps par la constitution de la liste des dangers entourant l'accès aux portes automatiques. Cette question a déjà été traitée dans le cadre du projet MODSafe [9] entre 2008 et 2012. Ce projet, initialement lancé par la Commission Européenne dans le cadre du 7ème "Research and Technological Development Framework Programme", avait pour but l'étude des différents standards de sécurité dans le domaine ferroviaire en Europe. Il a abouti à la rédaction de plusieurs rapports, dont une première analyse des dangers [30]. Dans ce document figure une liste des dangers les plus courants entourant les portes automatiques d'un train, résumés en figure 3.

A l'exception des risques 3.7 "électrocution" et 3.5 "Présence d'un passager entre deux wagons" qui sont hors du spectre de notre tâche, les autres risques se résument à différentes catégories de dangers. Tout d'abord, la mise en danger des passagers liée à la fermeture des portes (risques 3.2, 3.3 et 3.4) ; liée à un dysfonctionnement du comble-lacune (risque 3.6) ou à la chute d'un passager (risques 3.1 et 3.6).

B/ Priorisation des sous-tâches à traiter

Dès lors, on en déduit un ensemble de sous-tâches que les algorithmes développés doivent être en mesure d'effectuer :

Tout d'abord, la détection et le suivi des piétons aux alentours des portes pour estimer leur trajectoire dans un futur proche. L'objectif est de détecter d'éventuels déplacements de passagers dans la direction des portes peu avant leur fermeture afin de stopper leur fermeture ou encore d'émettre un signal indiquant aux piétons la fermeture imminente de la porte. Il s'agit ici d'une détection préventive des situations de risque impliquant la fermeture des portes sur des usagers et constitue un risque courant aux alentours des

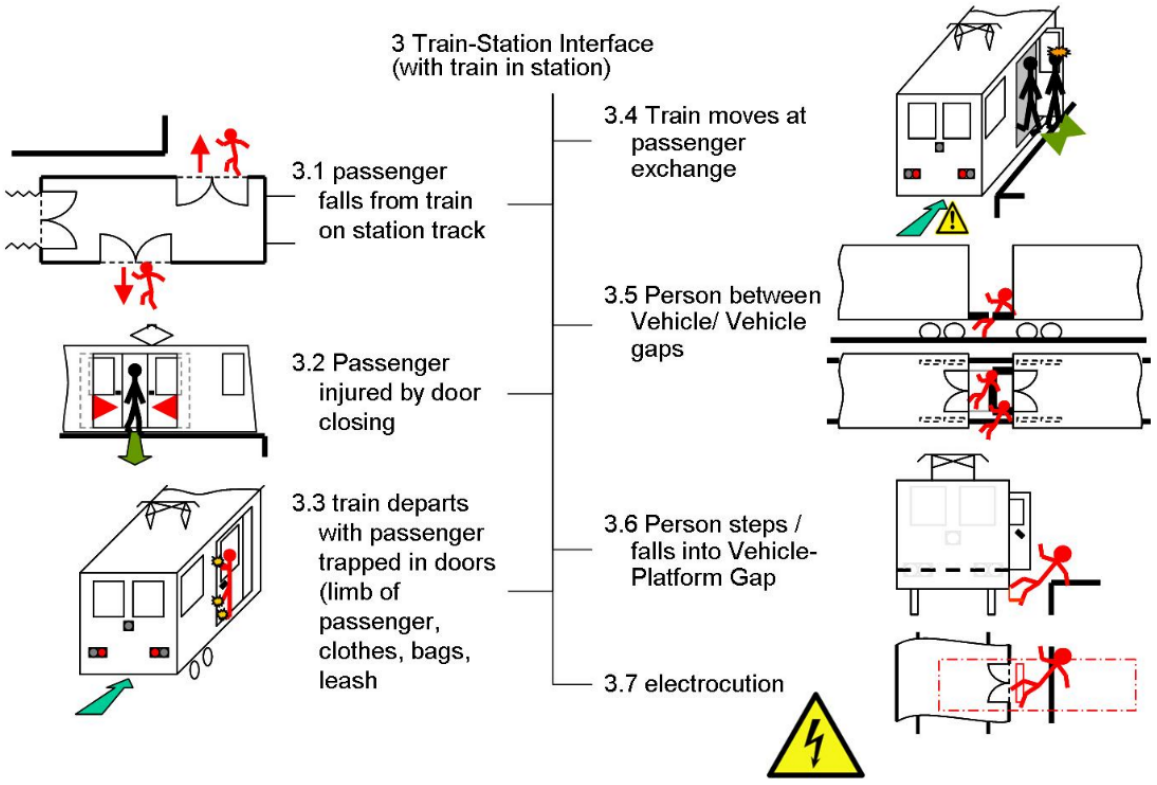


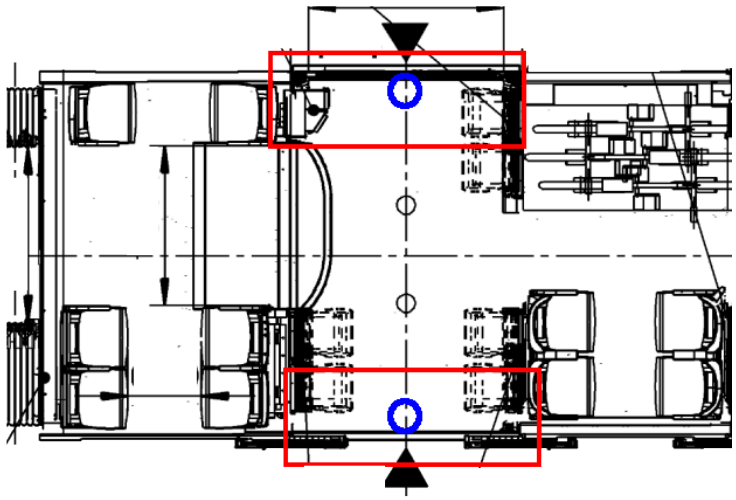
FIGURE 3 – Liste des dangers liés à l'interface train-quais. Source : [30]

portes automatiques. Ensuite, la détection de situations de mise en danger des usagers. Notamment la détection d'usagers, de membres d'usagers ou d'objets pris dans les portes, les usagers tombés au sol et nécessitant assistance ou encore le non déploiement du comble-lacune. Ces détections s'effectuent lorsque les usagers se trouvent déjà en situation de danger et visent à venir en aide aux usagers et éviter le sur-accident.

Dans le cadre du stage, nous nous sommes dans un premier temps intéressés à la détection préventive de passagers se dirigeant vers les portes automatiques peu avant leur fermeture.

C/ Matériel mis à disposition

Le matériel mis à disposition pour la détection de piétons est une paire de caméras fisheye situées sur le plafond du train, une en face de chaque porte du train (voir figure 4). Ces caméras possèdent un champ de vision panoramique. Les algorithmes développés seront donc basés sur du traitement d'images ou de vidéos sans son. Par ailleurs, d'autres équipements de détection sont disponibles sur les portes mais ne seront pas mis à contribution dans les algorithmes développés. On peut notamment citer des compteurs de passagers,



(a)

(b)

FIGURE 4 – Schéma d'un wagon du train autonome (4a) et image d'une caméra fish-eye (4b). En rouge figurent deux portes du wagon et en bleu les positions approximatives des caméras fish-eye.

quelques caméras internes et d'une caméra externe de faible qualité, actuellement utilisées par le conducteur pour assurer la sécurité dans les wagons et sur les quais ainsi que d'un laser externe pointé vers le comble-lacune pour valider son état de déploiement.

Sélection du type d'algorithme de suivi de piétons

A/ Etat de l'art des algorithmes de Multi Object Tracking

Division d'un algorithme de MOT en sous-tâches

La tâche décrite ici se résume à un suivi d'objets multiples connu dans la littérature sous le nom de Multi Object Tracking (MOT) ou suivi multi-objets et peut elle-même se décomposer en différentes sous-tâches. Pour chaque image d'une vidéo, les algorithmes de MOT doivent être en mesure de :

- Détecter les cibles présentes sur l'image. Dans notre cas, distinguer les piétons du fond de l'image et déterminer précisément leur position. En pratique, les cibles sont repérées par un cadre qui suit leur contours appelé bounding box.
- Associer à chaque cible un identifiant unique permettant de la suivre sur plusieurs images consécutives et ainsi reconstituer sa trajectoire.

Le suivi d'objets mobiles et particulièrement le suivi de piétons est en grand essor depuis le début des années 90, avec le nombre de papiers scientifiques traitant du sujet passant de quelques articles en 1990 à plus de 600 par an depuis 2016[28]. Il est particulièrement étudié pour ses nombreuses applications dans les domaines de la surveillance, de la robotique mobile ou encore des interactions homme-machine. Le suivi d'objet est une tâche complexe à cause de la grande variété de difficultés que les algorithmes de MOT doivent résoudre. Ces difficultés peuvent se résumer à des problèmes liés aux mouvements de caméra (changement d'inclinaison, zoom, flou cinétique...), à la variabilité de l'environnement (changement de luminosité, réflexions...), à la variabilité des cibles à suivre (vêtements des piétons, auto-occultation d'objets articulés...) ou encore à la consistance du traqueur dans le temps (occultation entre objets de même classe pouvant mener à une fin du suivi ou à des échanges d'identifiants entre les cibles).

Plus spécifiquement, dans le cas de notre application dans le domaine ferroviaire, on

s'attend à des difficultés supplémentaires :

— L'affectation des identifiants doit être très discriminante pour ne pas avoir d'échanges d'identifiants intempestifs.
— La forte densité de piétons lors de l'embarquement implique une grande occultation entre les piétons qui peut nuire au suivi.
— La contrainte du temps réel à cause de la criticité de la tâche traitée.

Le problème du suivi d'objets à été approché de différentes manières depuis les années 90. Dans le cadre de mon stage, je me suis concentré sur les méthodes traitant les images dans leur globalité. Je ne décrirai donc pas les méthodes traitant les images à l'échelle pixellique telles que celles se basant sur des détecteurs orientés détection de points d'intérêt comme les détecteurs de Harris[32] ou FAST[59] ou des descripteurs de points d'intérêt comme les algorithmes SIFT[44], SURF[25] ou ORB[52]. Les algorithmes que j'ai étudiés peuvent se distinguer en différentes sous-catégories non-exhaustives et non-exclusives. On parlera par la suite d'algorithmes de suivi "classiques" ou "neuronaux", par apprentissage "online" ou "offline" et basés sur du suivi par "identification" ou "association".

Distinction entre algorithmes de suivi "classiques" et réseaux neuronaux

Comme décrit dans l'étude de A. Brunetti et al. [28], la détection d'un objet prend place en trois étapes dans le cas d'un algorithme de détection "classique" : l'acquisition de l'image à traiter, l'extraction de caractéristiques et la classification des objets éventuels sur l'image. Dans le cas des algorithme neuronaux, l'extraction de caractéristiques est faite implicitement par les couches neuronales qui forment le réseau, voir figure 5. L'utilisation de ces techniques a considérablement grandi en popularité depuis l'arrivée des réseaux neuronaux convolutifs (Convolutional Neural Networks, ou CNN) qui sont devenus l'état de l'art parmi les algorithmes de détection [37], [55], [58], [35], [61] et dont l'utilisation croît dans le domaine du suivi d'objets multiples [40],[26] [38],[33], [36],[60]. Leur avantage principal est qu'ils épargnent aux data scientists la mise en place d'algorithmes d'extraction de caractéristiques qui peut s'avérer long et complexe. En revanche, les algorithmes neuronaux requièrent des quantités considérables de données d'entraînement et un certain temps d'entraînement. Ce qui explique que le récent développement des réseaux neuronaux a été déclenché par l'apparition de bases de données accessibles gratuitement (COCO[42] et ImageNet[53] pour la détection d'objets, MOT [39],[46], MARS [64] ou PETS [48] pour le suivi de piétons et la réidentification de piétons par exemple) et de réseaux pré-entraînés (voir transfer learning[5]). Pour notre application, nous implémenterons un traqueur basé sur des réseaux neuronaux. Ce qui nécessitera la recherche ou la création d'une base de données annotées.

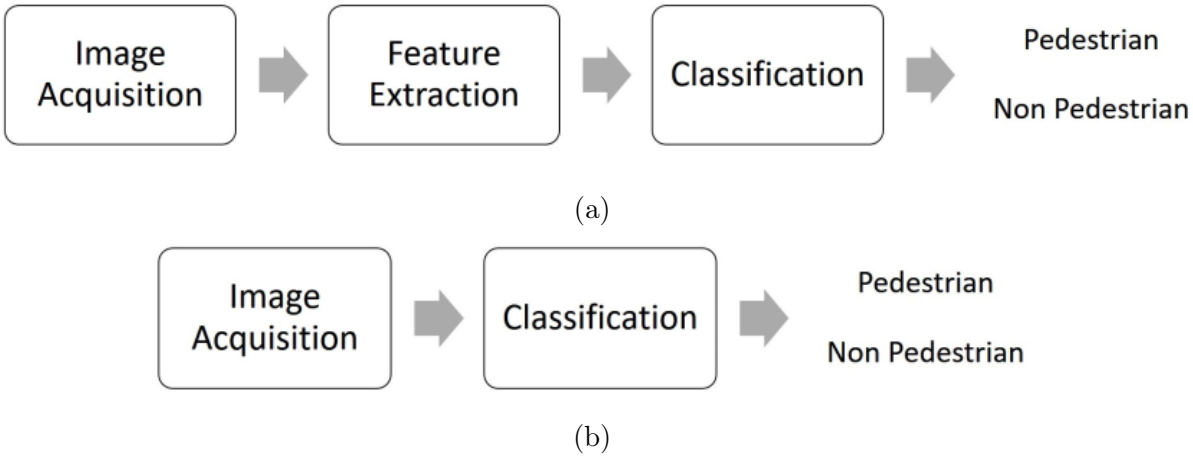


FIGURE 5 – Schéma récapitulatif des étapes de traitement de données dans le cas d'algorithmes classiques (5a) ou neuronaux (5b). Source : [28]

Distinction entre suivi par apprentissage online et offline

Un autre critère important dans le choix d'un traqueur pour notre application parmi ceux présents dans la littérature est leur méthode d'apprentissage. Il existe deux grandes familles de méthode d'apprentissage, appelées apprentissage en ligne (ou online) et hors ligne (ou offline). Les algorithmes de suivi à apprentissage offline ont besoin de données d'entraînement avant de pouvoir effectuer le suivi. Ils apprennent à distinguer les cibles les unes des autres et du fond de l'image à l'aide des données d'entraînement. Ces méthodes sont de ce fait incapables de suivre des cibles de classe non représentées dans les données d'entraînement, mais elles assurent une affectation d'identifiants discriminante entre cibles de même classe. A l'inverse, l'apprentissage "online" a lieu pendant le suivi. Le classificateur se spécialise dans la reconnaissance de la cible en étant exposé à différentes images de celle-ci. Ces méthodes ne requièrent pour ainsi dire aucun temps d'entraînement, mais sont sujettes à un phénomène de dérive : la perte de la cible à cause d'occultation peut amener le réseau à suivre un objet différent de la cible. Pour notre application, il semble complexe d'assurer la fiabilité des algorithmes par apprentissage online, surtout dans un contexte de cibles très dense comme dans notre cas. De ce fait, le traqueur sélectionné se base sur de l'apprentissage hors-ligne, entraîné sur des bases d'images de piétons. Un exemple de méthode par apprentissage online est la méthode STRUCK [31] dont une représentation est fournie en figure 6 ;

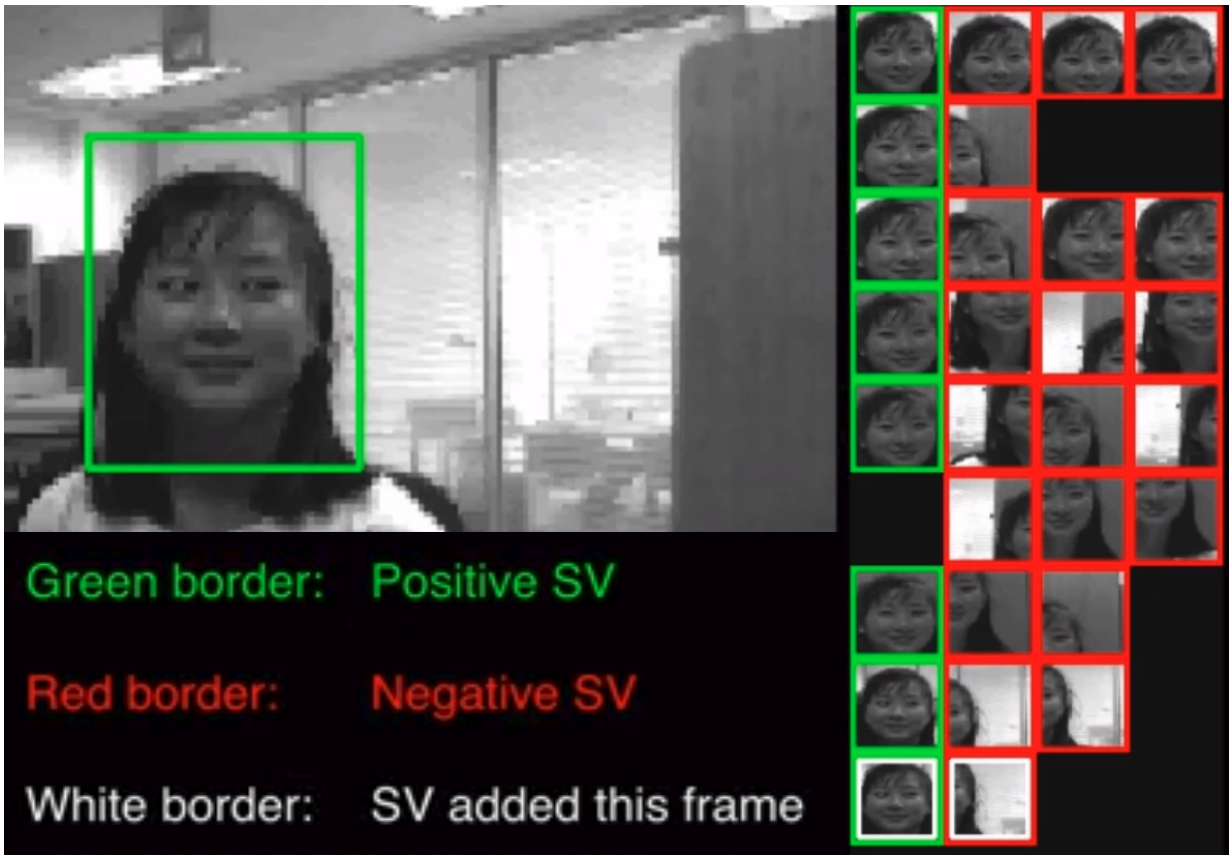


FIGURE 6 – Exemple d'utilisation de l'algorithme STRUCK, traqueur par apprentissage en ligne. Source : [18]. La méthode STRUCK est basée sur une Support Vector Machine (SVM). Les images sont recueillies par fenêtre glissante et un hard examples mining est appliqué. Des portions de l'image sont alors sélectionnées en exemples positifs (en vert, représentant la cible) et négatifs (en rouge, exemples ne représentant pas la cible mais suffisamment proches pour constituer de bons contre-exemples de la cible). Les vecteurs caractéristiques calculés à partir de chaque image représentent alors une projection des images dans un espace de dimension finie dans lequel une frontière est calculée par le SVM. De ce fait, à chaque nouvelle image ajoutée au pool d'images, les poids du SVM sont recalculés, ce qui revient à une modification en ligne du classificateur du réseau.

Distinction entre suivi par identification et par association

Enfin, nous nous inspirerons de la distinction effectuée dans le recueil [45] entre algorithmes de suivi par identification et de suivi par association (appelés respectivement "detection-free tracking" et "detection-based tracking" dans [45]) pour distinguer les algorithmes de suivi présents dans la littérature. Dans le cas du suivi par identification, la cible est détectée lors de sa première apparition et converti en un vecteur caractéristique. L'étape de suivi consiste alors lors de l'acquisition d'une nouvelle image en la localisation de la cible dans le voisinage proche de la détection précédente. Le vecteur caractéristique extrait de cette manière est alors comparé à celui de la cible. Ces techniques ont pour avantage d'avoir un temps d'exécution très faible, puisqu'elles consistent en une simple inférence d'un algorithme d'extraction de caractéristiques. Mais elles requièrent autant d'inférences que de cibles suivies. Cela est handicapant dans le cadre de notre application où on s'attend à un nombre important de cibles simultanées, et où la contrainte du temps réel est de nécessité absolue. Un exemple de méthode de suivi par identification est les réseaux dits 'siamois' (voir [40],[26] [38],[33] [36]). Dans le cas du suivi par association, à chaque image, un détecteur fournit l'ensemble des cibles en une unique inférence. Un algorithme d'association se charge alors de coupler l'ensemble des cibles de l'image précédente aux cibles de la nouvelle image en minimisant une métrique donnée. Un schéma récapitulatif de la distinction entre suivi par identification et suivi par association est disponible en figure 7.

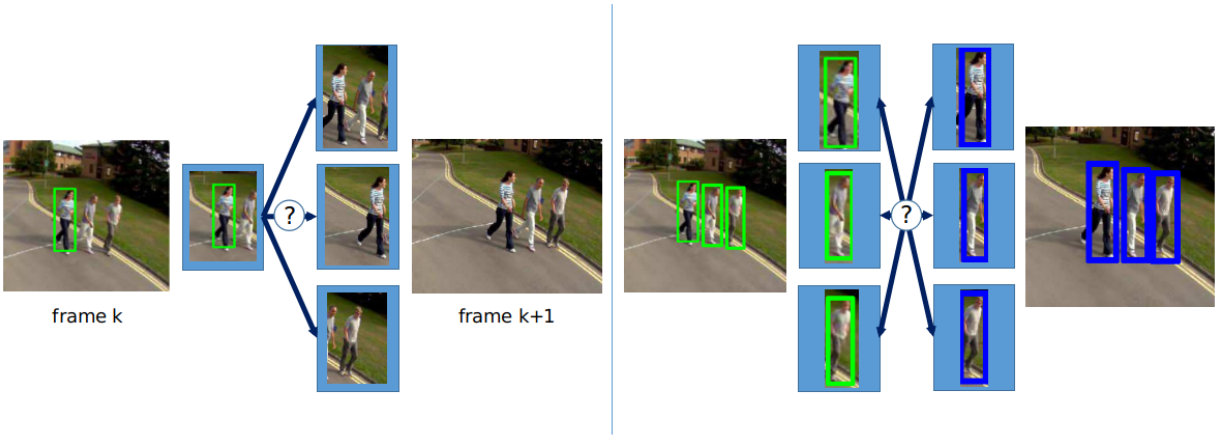


FIGURE 7 – Schéma récapitulatif de la distinction entre suivi par identification (gauche) et suivi par association (droite). Dans le cas du suivi par identification, pour chaque cible, on recherche sur l'image suivante l'ensemble de pixels les plus ressemblant à la cible. Le suivi par association quant à lui est constitué d'une étape de détection sur la nouvelle image puis d'une étape d'association entre les bounding boxes précédentes et actuelles. L'identification peut alors être vue comme une recherche de 1 parmi N répétée pour N cibles, l'association est une recherche de N parmi N effectuée une seule fois.

B/ Description chaîne de traitement finale

On en conclut que pour notre application, nous utiliserons un algorithme de suivi de piétons appartenant au sous-ensemble des algorithmes par réseau neuronal, par apprentissage hors-ligne et de suivi par association. On en déduit alors un squelette de chaîne de traitement résumé en figure 8. Cette chaîne de traitement peut se diviser en deux chaînés dites "chaîne d'inférence" et "chaîne d'évaluation". La chaîne d'inférence est linéaire. Les images du dataset d'images (dataset d'images de suivi de piétons) sont fournies à un détecteur qui se charge de détecter l'ensemble des piétons présents. Le détecteur fournit pour chaque cible une annotation qui est une bounding box encadrant la cible associée à la classe "personne". Ces annotations, et optionnellement les images, sont alors fournies au tracker qui va associer un identifiant unique à chaque bounding box, formant ainsi les trajectoires des piétons. Une itération de la chaîne d'inférence génère donc l'ensemble des bounding boxes des cibles présentes sur une image et leur associe un identifiant unique. La chaîne d'inférence est donc la chaîne qui serait implémentée sur le système embarqué final. Dans notre cas, nous cherchons à quantifier l'efficacité de l'algorithme de suivi et à visualiser les résultats sur les images traitées. Pour ce faire, nous avons rajouté une chaîne d'évaluation constituée de trois briques : une étape de prétraitement , une métrique d'évaluation de l'étape de détection ainsi qu'une métrique d'évaluation de l'étape de tracking. De plus, nous avons rajouté deux briques de conversion en vidéos des résultats de détection et de tracking (non représentées ici). Par la suite, nous décrirons les algorithmes choisis pour chaque brique algorithmique ainsi que le dataset d'images de suivi de piétons sélectionné.

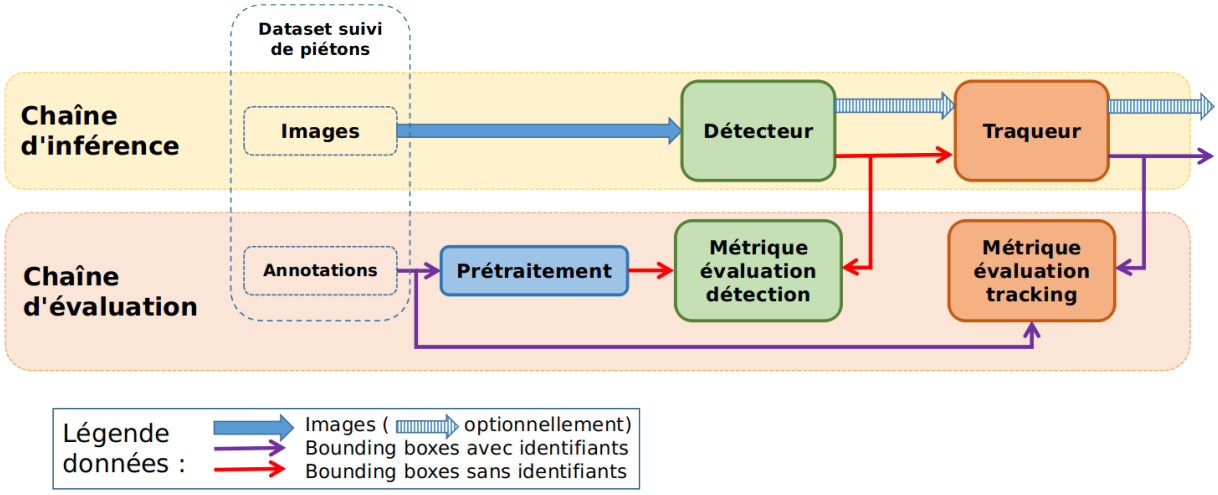


FIGURE 8 – Schéma récapitulatif de la chaîne de traitement implémentée. Les flèches représentent la transmission de données entre les briques algorithmiques.

Mise en place de l'implémentation

A/ Mise en place de la chaîne d'inférence

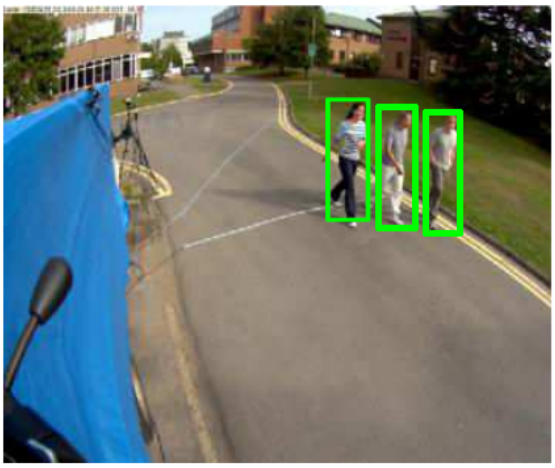
Choix de la brique de détection

Pour rappel, la tâche de détection consiste en la distinction des cibles du fond de l'image, en leur encadrement par une bounding box et l'affectation de la classe "piéton". De nombreux détecteurs par réseaux neuronaux convolutifs ont été développés depuis quelques années et le choix parmi ceux-ci revient à un compromis entre la qualité de détection (au sens de la précision et du rappel, voir plus loin) et le temps d'inférence de ces réseaux. Parmi les réseaux couramment utilisés récemment, on peut citer :

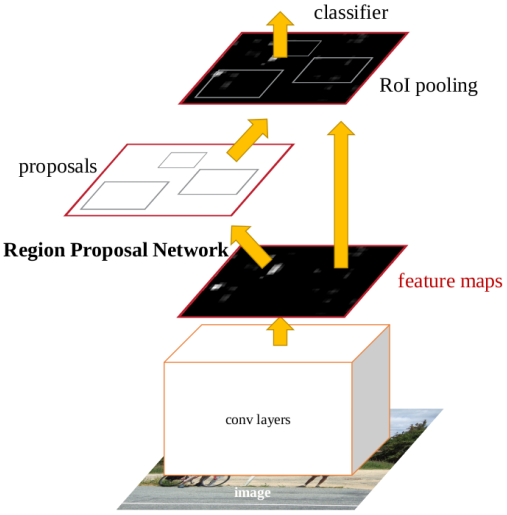
— Les détecteurs en deux étapes comme Faster R-CNN[51] (voir schéma 9b). L'image traitée est tout d'abord convertie en "feature maps" par une extraction de caractéristiques effectuée par un réseau neuronal convolutif appelé couramment le "backbone" du détecteur. Vient alors une étape de pré-détection de régions d'intérêts sur les features maps par le Region Proposal Network (RPN) qui fournit un ensemble de bounding boxes possédant potentiellement des cibles. Enfin, un classificateur se charge d'affilier une classe à chaque région d'intérêt. De plus amples détails sur l'implémentation de Faster R-CNN sont fournis en annexe *Précisions sur le fonctionnement de Faster R-CNN.*

— Les détecteurs en une étape comme SSD[43], YOLO[50] ou encore RetinaNet[41]. Dans ce cas, les étapes de pré-détection de régions d'intérêt et de classification sont fusionnées en une seule étape de détection gérée par un seul réseau neuronal.

D'une manière générale, les détecteurs en deux étapes offrent une meilleure qualité de détection au prix d'un temps d'inférence plus important que les détecteurs en une étape. Pour notre implémentation, nous avons utilisé dans un premier temps le détecteur Faster R-CNN avec pour backbone le réseau ResNet101. La librairie utilisée pour l'implémentation du détecteur est la Tensorflow Object Detection API (TFODAPI)[23], une extension de la librairie Tensorflow utilisée dans le domaine du deep learning (voir annexe *Précisions sur Tensorflow Object Detection API (TFODAPI)* pour plus de détails).



(a)



(b)

FIGURE 9 – Exemple de détection de piétons (9a) et schéma de fonctionnement du détecteur Faster R-CNN (9b). Source : [51]

Choix de la brique de tracking

L'étape de suivi, qui consiste en l'affectation d'un identifiant unique à chaque cible préalablement détectée, est effectuée par l'algorithme du Simple Online Realtime Tracking (SORT) [27] qui était classé lors de sa création en 2017 comme le meilleur traqueur Open Source disponible[19]. La motivation de la création du traqueur SORT était de concevoir un traqueur le plus simple possible pouvant fonctionner en temps réel. Là où la plupart des traqueurs développés se basent sur l'apparence des cibles pour les retrouver sur différentes frames, SORT relègue la qualité de l'identification au détecteur en amont et ne considère que les dimensions et positions des bounding boxes entre deux frames successives pour effectuer son suivi. Chaque bounding box est décrite par un vecteur d'état $x = [u, v, s, r, u', v', s']^T$ où (u, v) désigne le centre de la bounding box, $s = w * h$ est son aire et $r = w/h$ son rapport avec w et h respectivement sa largeur et sa hauteur (voir 10a) et u', v' et s' les dérivées temporelles de u, v et s. Pour l'ensemble des bouding boxes d'une frame, on suppose que la vitesse de la cible dans le repère image est constante entre deux frames consécutives et le bruit sur les variables du vecteur d'état est supposé gaussien.

Le principe de fonctionnement du SORT se base sur un filtre de Kalman prédictif sur les vecteurs d'état des bouding boxes avec hypothèse de vitesse constante des cibles dans le repère image.

Une itération de SORT se divise en 4 étapes distinctes résumées en figure 10b et présentées sommairement ci-après :

— On considère un état initial dans lequel l'ensemble des bounding boxes conservées ont été affiliées à la trajectoire d'une cible unique. La première étape de SORT consiste en une réception des bounding boxes nouvellement détectées par le détecteur sur la nouvelle frame (désignées par la suite par "bounding boxes actuelles"), ce qui correspond à l'étape d'observation du filtre de Kalman prédictif.
— Par la suite, en supposant que la vitesse des cibles est restée inchangée entre les frames précédente et actuelle, la bouding box estimée de chaque cible est prédite dans la nouvelle frame à l'aide d'un schéma d'Euler. Ce qui correspond à l'étape de prédiction du filtre de Kalman prédictif.
— Vient alors l'étape d'association entre les bounding boxes estimées et actuelles. Cette étape est effectuée à l'aide de la méthode Hongroise[15] qui consiste en une maximisation du poids de l'ensemble des couples des bouding boxes estimées et actuelles par une sélection gloutonne des meilleurs couples de bounding boxes. Le poids pour un couple de bounding boxes étant défini comme leur intersection divisée par leur union (métrique couramment appelée Intersection Over Union (IOU), voir plus loin). Enfin, les couples résultants de cette association possédant une IOU inférieure à une valeur seuil sont rejetés car considérés comme non-concluants.
— En dernier vient l'étape de correction des bounding boxes. Chaque couple de bounding boxes représente théoriquement une cible identique. Leurs vecteurs d'état peuvent alors être moyennés pour obtenir une meilleure estimation de la cible. Les bounding boxes corrigées ainsi calculées sont alors concaténées aux trajectoires de leurs cibles respectives.

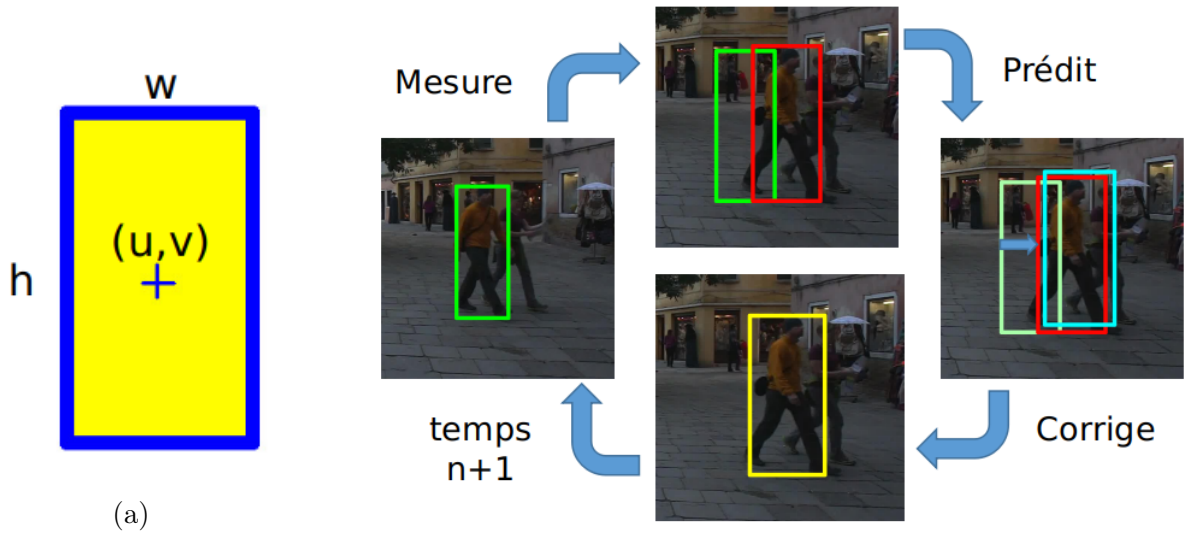
De plus amples détails sur les paramètres du filtre de Kalman et sur les étapes de SORT sont disponibles en annexe *Précisions sur le filtre de Kalman employé pour SORT*.

Pendant ce cycle, plusieurs cas particuliers peuvent se présenter :

Dans le cas où une bounding box estimée ne possède pas d'équivalent parmi les bounding boxes actuelles, la cible est considérée comme perdue et la bouding box estimée devient la bouding box corrigée.
Dans le cas où une bounding box actuelle ne possède pas d'équivalent parmi les bounding boxes estimées, la bounding box actuelle constitue le début de la trajectoire d'une nouvelle cible. La bouding box actuelle devient alors la bounding box corrigée et est ajoutée à une trajectoire vierge.

Deux paramètres sont introduits pour gérer ces cas particuliers :

L'age maximum d'une trajectoire perdue (MAXAGE) qui correspond à la durée maximale de prédiction d'une trajectoire lorsque celle-ci est perdue avant de la considérer comme définitivement terminée.
L'age minimal d'une trajectoire (MINHITS) qui correspond au nombre minimal de bounding boxes consécutives nécessaire avant de considérer une trajectoire comme commencée.



(a) (b)

FIGURE 10 – (10a) Schéma représentatif d'une bounding box , où (u,v) représentent son centre, w sa largeur et h sa hauteur. (10b) Schéma simplifié des 4 étapes de SORT. Les cadres vert, rouge, cyan et jaune représentent respectivement les bounding boxes précédente, actuelle, estimée et corrigée.

L'implémentation de SORT utilisée est l'implémentation du papier original [27] fournie sur github[19] codée en Python et reposant sur la librairie filterpy[6] pour l'instanciation d'un filtre de Kalman. Nous étudierons par la suite les effets de ces paramètres sur la qualité du suivi.

B/ Mise en place de la chaîne d'évaluation

Sélection de la métrique d'évaluation pour l'étape de détection

Pour évaluer l'efficacité de notre détecteur, il nous faut définir un ensemble de métriques propre à la tâche de détection. Nous avons sélectionné les métriques COCO (Common Objects in COntext) tirées du concours du même nom (voir [42]). Dans le cadre de la détection, deux métriques sont couramment utilisées : la précision et le rappel. Dans les éditions plus modernes du concours COCO, ces deux métriques se déclinent en des variantes plus complexes décrites en annexe *Retour sur les métriques COCO*. Mais, bien que les résultats que nous présenterons en partie C/ se basent sur ces variantes plus complexes, nous simplifierons la description des métriques de détection à ces deux métriques simples

que sont la précision et le rappel.

Considérons un ensemble d'éléments à classifier selon deux classes "vrai" ou "faux" où la classe "vrai" est la classe d'intérêt tandis que la classe "faux" représente l'absence de la classe "vrai". Lors de l'application d'un classificateur, certains éléments sont sélectionnés (ceux recevant la classe "vrai") tandis que les autres sont mis de côté (ceux recevant la classe "faux"). Cela amène à la création de 4 ensembles d'éléments résumé par le tableau 11.

		Sélection	
		Sélectionné	Non Sélectionné
Pertinence	Vrai	Vrai Positif	Faux Négatif
	Faux	Faux Positif	Vrai Négatif

FIGURE 11 – Tableau récapitulatif des 4 cas possibles lors de l'application d'un classificateur à un ensemble d'éléments à traiter.

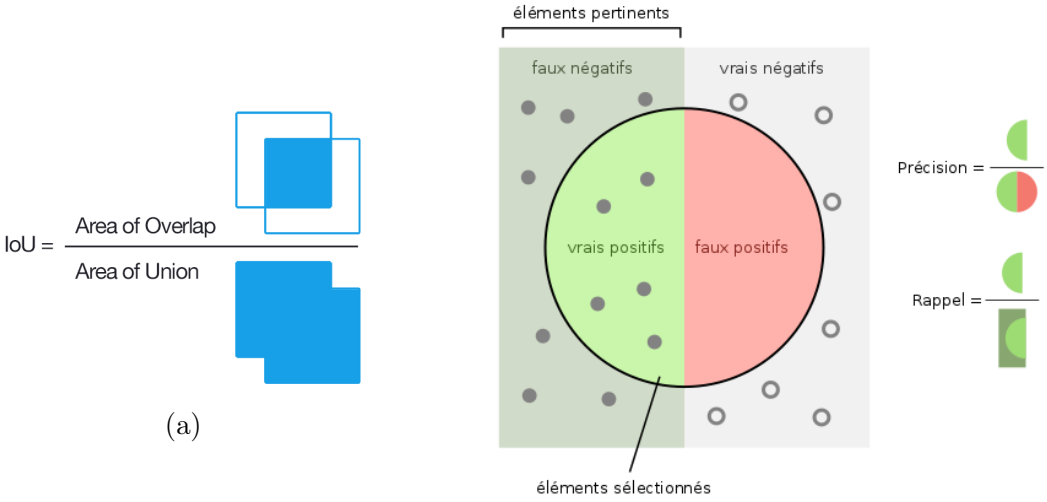
Dès lors, le rappel et la précision dans le cas d'une tâche de classification sont définis de la manière suivante :

La précision est la proportion d'éléments pertinents parmi ceux sélectionnés. Soit le nombre de vrai positifs divisé par le nombre d'éléments sélectionnés.
Le rappel d'un échantillon est la proportion d'éléments pertinents sélectionnés. Soit le nombre de vrais positifs divisé par le nombre d'éléments pertinents.

Voir le schéma 12b.

Or dans le cas présent, nous cherchons à qualifier l'efficacité d'un détecteur, donc de la classification ainsi que du positionnement d'éléments dans une image. Pour remettre ces deux métriques dans le cadre d'une tâche de détection, on définit un élément vrai ou pertinent comme étant une bounding box appartenant à la vérité terrain des annotations de piétons. Lorsqu'un détecteur fournit un ensemble de bounding boxes pour une image, chaque bouding box est comparée à l'ensemble des bouding boxes pertinentes. Une bounding box pertinente est "sélectionnée" s'il existe une bounding box fournie par le détecteur possédant un IOU supérieur à une valeur seuil (voir 12a). Ces métriques sont fournies par Tensorflow Object Detection API (TFODAPI), voir annexe *Précisions sur Tensorflow Object Detection API (TFODAPI)* pour plus de détails.

A ces deux métriques, on rajoute le taux de Frames Per Seconds du détecteur (FPS détecteur) pour rendre compte du temps d'inférence par image du détecteur.



(a) (b)

FIGURE 12 – Schéma récapitulatif de l'Intersection Over Union (IOU) (12a). Schéma récapitulatif des concepts de Rappel et de Précision (12b). Le rappel d'un échantillon peut se résumer en la proportion d'éléments pertinents sélectionné tandis que la précision est la proportion d'éléments pertinents parmi ceux sélectionnés. Source : [13].

Sélection de la métrique d'évaluation pour l'étape de tracking

Pour quantifier l'efficacité du traqueur, nous avons utilisé les métriques CLEAR[57] couramment utilisées dans le domaine du suivi multi-objets. Nous ne préciserons ici que les métriques que nous avons utilisé, les autres sont décrites dans [46].

Lors de la constitution de trajectoires, différentes erreurs d'association peuvent survenir. Ces erreurs se résument dans le cas des métriques CLEAR au taux de Faux Positifs (FP), Faux Négatifs (FN), échanges d'identifiants (ou ID switch, IDsw) et aux fragmentations de trajectoire (Frag) comme présenté dans le schéma 13.

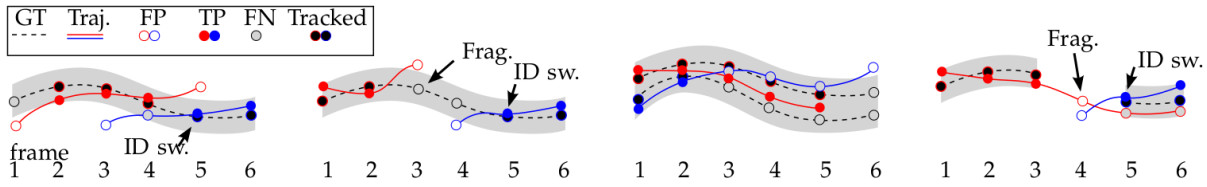


FIGURE 13 – Schéma récapitulatif des différentes erreurs d'association de bounding box à une trajectoire. Source : [46].

Les courbes en pointillé ou en trait plein représentent des trajectoires, véritables ou estimées composées de bounding boxes symbolisées par des points. Ces trajectoires sont représentées dans un espace dans lequel l'axe des abscisses est discret et représente les

numéros de frame et les ordonnées représentent la similarité entre des bounding boxes. En pointillés sur ce schéma figure la trajectoire véritable de la cible telle que présente dans les annotations de la vérité terrain (ou Ground Truth, GT). La zone en gris caractérise l'ensemble des coordonnées de bounding box possédant une IOU avec la bounding box véritable supérieure à la valeur seuil d'association. Les courbes continues rouge et bleues sont des trajectoires estimées par un traqueur. A chaque frame, les bounding box peuvent être associées à une trajectoire (points pleins) ou non (point creux). Une bounding box de la trajectoire GT n'étant associée à aucune trajectoire estimée génère un Faux Négatif (FN). A l'inverse, une bouding box d'une trajectoire estimée n'étant associée à aucune bouding box parmi les trajectoires GT génère un Faux Positif (FP). Enfin, toute discontinuité d'une trajectoire estimée génère une Fragmentation (Frag) et tout échange de trajectoire génère un ID switch (IDsw).

Le calcul du nombre de ces erreurs prises séparément constituent des métriques à part entière. Mais plutôt que de traiter ces différentes métriques indépendamment, il est préférable d'utiliser une métrique composite appelé Multi-Object Tracking Accuracy (MOTA) et définie par l'équation 1.

$$MOTA = 1 - \frac{\sum_t (FN_t + FP_t + IDsw_t)}{\sum_t GT_t} \tag{1}$$

La MOTA reflète la consistance du traqueur dans le temps. Elle diminue lorsque le nombre de faux positifs, faux négatifs ou échanges d'identifiants augmente et peut prendre des valeurs négatives. Dans notre cas, on la multipliera par 100 afin d'obtenir un résultat maximal de 100. A celle-ci vient s'ajouter la Multi-Object Tracking Precision (MOTP) définie comme l'IOU moyen des bounding boxes traquées et permettant de rendre compte de la précision moyenne des contours des bounding boxes.

De plus, on retrouve parmi les métriques CLEAR les notions de rappel et précision comme définies pour les métriques COCO. La raison étant qu'un traqueur peut comme nous aurons l'occasion de le voir générer ou supprimer de fausses détections et ainsi influencer sur ces deux valeurs.

Par la suite, nous utiliserons en supplément la moyenne harmonique de la précision et du rappel, appelée métrique F1, afin de rendre compte de l'évolution moyenne de ces deux critères.

L'ensemble de ces métriques sont fournies par la librairie pymotmetrics[17] qui est compatible avec les résultats du concours MOT.

Enfin, à ces métriques nous rajoutons le taux de Frames Per Seconds du traqueur (FPS traqueur) pour rendre compte du temps d'inférence par image du traqueur.

Un résumé de ces métriques est fourni en figure 14.

La chaîne de traitement une fois complétée est résumée en figure 15.

	Rappel	Précision	F1	MOTA	MOTP	FPS
Mieux	Augmente	Augmente	Augmente	Augmente	Augmente	Augmente
Meilleur	100%	100%	100%	100	100	+Inf.

FIGURE 14 – Tableau récapitulatif des différentes métriques de tracking sélectionnées. Pour toutes ces métriques, une augmentation de leur valeur implique une amélioration du suivi, et leurs valeurs maximales sont données.

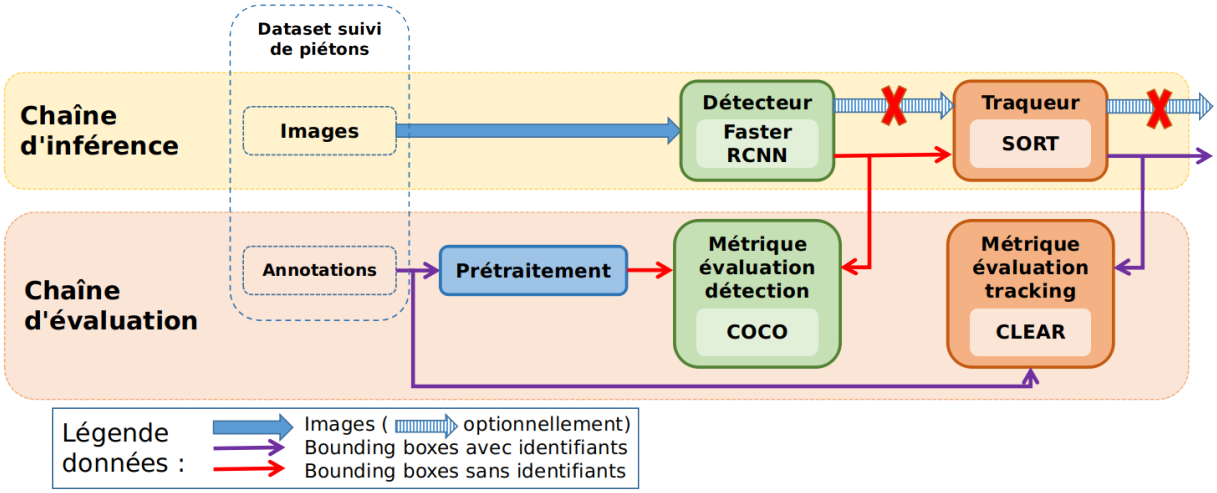


FIGURE 15 – Schéma récapitulatif de la chaîne de traitement complétée. Dans notre cas, le traqueur ne nécessite pas les images pour fonctionner, juste les annotations fournies par le détecteur.

Par ailleurs, les superpositions des annotations sur les images MOT sont effectuées à l'aide de la librairie OpenCV[12]. Et la concaténation des images en vidéo est réalisée avec ffmpeg[11].

C/ Sélection d'un dataset d'images de suivi de piétons

La base de données utilisée pour l'entraînement et la validation de nos algorithmes de détection et de suivi est le dataset MOT (Multi-Object Tracking) [39], [46]. Ce dataset est utilisé dans le contexte d'un concours appelé le MOT Challenge[10]. L'ensemble des images et annotations utilisées est la combinaison des données des éditions 2017 et 2019 du concours. Les annotations fournies sont adaptées au suivi de piétons : pour chaque frame sont fournies l'ensemble des bounding boxes des piétons et d'objets pouvant les occulter. A chaque bounding box est associée un identifiant de classe et un identifiant d'instance. Les frontières des bounding boxes suivent les contours des cibles telles qu'elles seraient vues

sans aucune occultation. Cela permet d'avoir dans les annotations l'ensemble des bounding boxes d'une cible donnée quand bien même celle-ci serait partiellement ou temporairement occultée. Des guides décrivant le format de données utilisé pour les annotations MOT sont disponibles sur leur site : [10]. L'ensemble des étapes de prétraitement des annotations MOT pour l'entraînement du détecteur sont fournies en annexe *Chaîne de prétraitement des données MOT*. Les données fournies dans le cadre du concours ont pour avantage de représenter des piétons dans un environnement urbain avec une grande variété d'environnements, de conditions de luminosité et de densité de personnes.

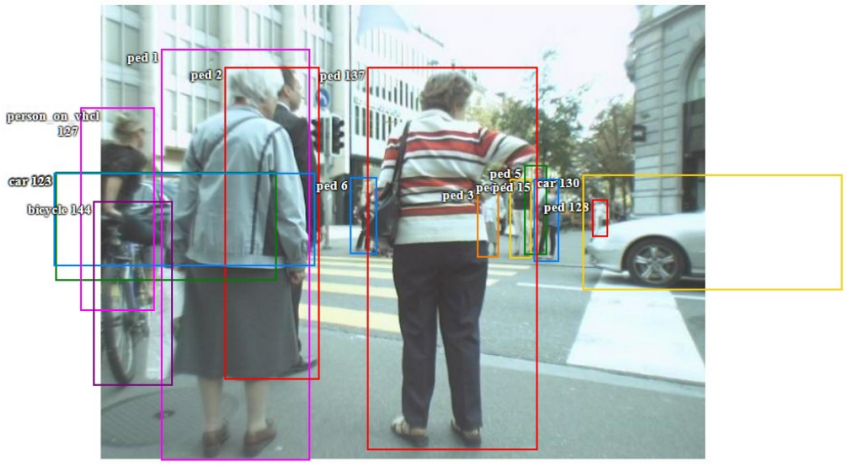


FIGURE 16 – Exemple d'annotations MOT rajoutée sur une image du dataset. Source : [46].

Résultats expérimentaux

A/ Démarche expérimentale

Pour le développement de notre traqueur, nous avons voulu étudier l'influence du détecteur et du traqueur sur les résultats de suivi finaux. Pour cela, nous avons constitué deux chaînes de traitement. Une chaîne dite "témoin" formée d'un détecteur utilisé avec les paramètres par défaut fournit par TFODAPI. Et une autre dite "entraînée" dont le détecteur a été ré-entraîné sur un set de données particulier. Les paramètres de ces deux chaînes sont fournis ci-après dans la table 1

Paramètres	**Détecteur témoin**	**Détecteur entraîné**
Structure	Faster R-CNN	Faster R-CNN
Backbone	ResNet101	ResNet101
Dataset d'entraînement	COCO14	COCO14 + MOT
Nombre d'échantillons maximum en sortie du RPN	300	100

TABLE 1 – Tableau récapitulatif des paramètres des détecteurs témoin et entraîné

Le set de données utilisé a été constitué à partir des données MOT des éditions 2017 et 2019. Les vidéos constituant ces datasets ont été classées selon la densité de personnes visibles en données "faciles", "moyennes" et "difficiles". Deux sets de données, un set d'entraînement et un set de test ont alors été constitués et ont été équilibrés en fonction de la difficulté des vidéos, de l'inclinaison et du mouvement de la caméra et de la luminosité ambiante. La répartition des données dans les différents sets ainsi que des informations complémentaires des différentes vidéos sont fournies en annexe *Chaîne de prétraitement des données MOT*.

La première étape de notre démarche expérimentale a été la constitution d'une chaîne de traitement témoin constituée d'un détecteur pré-entraîné sur le dataset COCO. Ce dataset contient 81 classes différentes dont la classe "person" qui est celle que nous cherchons à

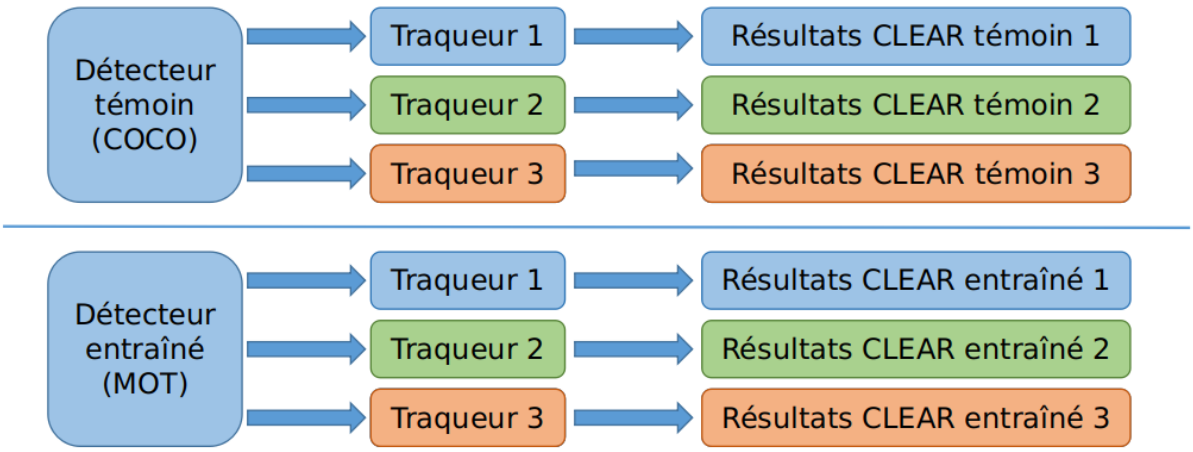


FIGURE 17 – Schéma récapitulatif des 6 chaînes de traitement testées. Le traqueur 1 correspond à la configuration par défaut de SORT telle que décrite dans [27]

détecter.

Les deux détecteurs "témoin" et "entraîné" ont été testés selon la métrique COCO sur les données "faciles" de l'ensemble des données MOT de test. Le set de données utilisé pour l'entraînement de la chaîne entraînée est le set de données "faciles" des données MOT d'entraînement. L'ensemble des étapes de prétraitement appliquées aux données MOT pour l'entraînement du détecteur sont décrites en annexe *Chaîne de prétraitement des données MOT*.

Chacune de ces chaînes de traitement a été testée selon 3 configurations du traqueur à l'aide des métriques CLEAR, aboutissant à la création de 6 chaînes de traitement résumées en figure 17. Les trois configurations du traqueur sont trois couples de valeurs pour les paramètres MAXAGE et MINHITS précisés en partie A/. Ces trois configurations sont résumées dans le tableau 2. La première configuration du traqueur dit "témoin" est la configuration telle que décrite dans le papier original du traqueur SORT (voir [27]).

	Valeur MAXAGE	Valeur MINHITS
Configuration traqueur "témoin"	1	3
Configuration 2	5	3
Configuration 3	5	6

TABLE 2 – Tableau récapitulatif configurations du traqueur

B/ Résultats des chaînes de traitement basées sur le détecteur témoin

Résultats de détection du détecteur témoin selon les métriques COCO

Pour rappel, le détecteur témoin est un Faster R-CNN ResNet101 pré-entraîné sur COCO14. Il a été testé sur les données MOT faciles de test et a fourni les résultats suivants :

— Résultat en précision : mAP@0.5 = 0.67
— Résultat en rappel : mAR@100 = 0.51

Ce qui peut s'interpréter selon les deux affirmations suivantes :

— Un tiers des annotations détectées au seuil d'IOU de 0.5 sont erronées.
— La moitié des annotations n'ont pas été détectées au seuil d'IOU de 0.5.

A noter que bien que le détecteur témoin génère des résultats selon les 81 classes du dataset COCO sur lequel il a été entraîné, les résultats fournis ici sont calculés uniquement sur les annotations de personnes. Afin de comprendre l'origine de ces forts taux d'erreur, les bounding boxes détectées par le détecteur ont été superposées sur les images de MOT dont quelques exemples sont fournis en figure 18. Après analyse des détections, plusieurs types d'erreurs se dégagent.

Concernant l'erreur en précision, on constate que lorsque des personnes se rapprochent trop de l'objectif de la caméra, leur bounding box se divise souvent en bounding boxes de taille inférieure (voir l'image de gauche en figure 22). Ces erreurs peuvent s'expliquer par la différence de dimensions des objets à détecter par rapport aux dimensions de l'image entre les images du dataset COCO, sur lesquelles le détecteur témoin a été entraîné, et celles de MOT, sur lesquelles il est testé. Le détecteur habitué à détecter des cibles de faibles dimensions aura tendance à détecter plusieurs cibles lorsque confronté à une cible de grandes dimensions. Par ailleurs, certains objets comme des vêtements, des écharpes ou des statues sont détectées à tort comme des personnes, ce qui génère des fausses détections. Enfin, des parties d'image avec de fortes densité de piétons sont parfois annotées à tort comme étant une unique entité (voir l'image de gauche en figure 23).

Concernant l'erreur en rappel, on constate que les personnes trop occultées ou trop éloignées de l'objectif de la caméra sont rarement détectées.

Résultats de suivi du traqueur témoin selon les métriques CLEAR

A ce détecteur témoin on vient alors rajouter le traqueur SORT avec les paramètres par défaut tels que décrit dans le papier original du traqueur SORT[27]. Les paramètres

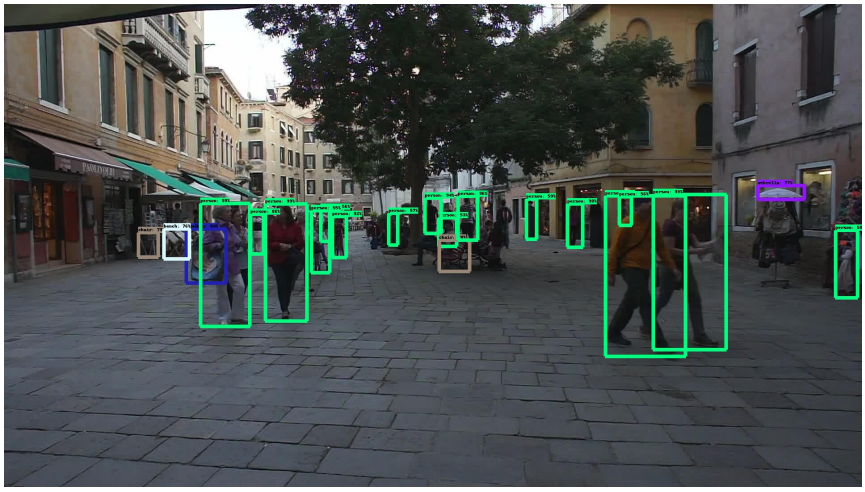


FIGURE 18 – Résultat de la superposition des bounding boxes fournies par le détecteur témoin sur l'image du dataset MOT correspondante. La couleur des bboxes correspond à la classe de l'objet détecté, le dataset COCO sur lequel a été entraîné le détecteur témoin contient 81 classes dont la classe "personne" ici représentée en vert. On constate par ailleurs que la bounding box la plus à droite identifie à tort une écharpe comme étant une personne.

du traqueur SORT sont MAXAGE = 1 et MINHITS = 3.

Les résultats obtenus selon les métriques CLEAR sur les données MOT faciles et moyennes sont celles fournies en figure 19.

Rappel	Precision	F1	MOTA	MOTP	FPS détecteur	FPS tracker
46.9	**83.7**	**59.8**	**36,1**	**75,7**	**6,67**	**101,8**

FIGURE 19 – Résultats de suivi de la chaîne de traitement constitué du détecteur témoin et du traqueur témoin selon les métriques CLEAR.

On trouve une faible valeur de rappel et une valeur correcte de précision. Ces valeurs nous serviront de valeur de base dans la suite du rapport. Par ailleurs, le MOTA, qui traduit la consistance du traqueur dans le temps et le nombre de fausses annotations, est très faible. Sa faible valeur s'explique par un grand nombre de fausses annotations, dûes en grande partie au détecteur, et aux échanges d'identifiants. Dans la configuration témoin du traqueur, L'age maximal d'une trajectoire perdue est de 1 (MAXAGE=1). Ce qui signifie qu'une trajectoire est considérée comme perdue dès qu'une occultation apparaît. Ce qui peut expliquer le grand nombre d'échanges d'identifiants. Quant au MOTP, il est de valeur moyenne, de 75,7 pour une valeur comprise entre 50 et 100. Enfin, les résultats en FPS du détecteur et du traqueur nous indiquent que le détecteur est le goulet d'étranglement de notre chaîne de traitement en terme de temps de traitement. Le traqueur possédant une

vitesse de traitement amplement suffisante pour notre application.

La superposition des bounding boxes sur les images donne une image comme celle présentée en figure 20. La couleur de chaque bounding box correspond à l'identifiant d'instance qui lui a été attribué. Cette couleur est tirée aléatoirement lors de la première détection de la personne (lorsque le nombre de bounding boxes associées à la même cible consécutives est supérieur à MINHITS) et reste inchangée tant que le suivi n'est pas interrompu. Le nombre de couleurs possibles étant limité, il est possible que plusieurs individus d'identifiant d'instance différents possèdent une bounding box de même couleur.

En comparant les vidéos générées à partir des détections fournies par le détecteur et des annotations en sortie du traqueur, on s'aperçoit aussi d'une stabilisation des bounding boxes dans le temps, résultat de la correction des bounding boxes par le filtre de Kalman du traqueur.

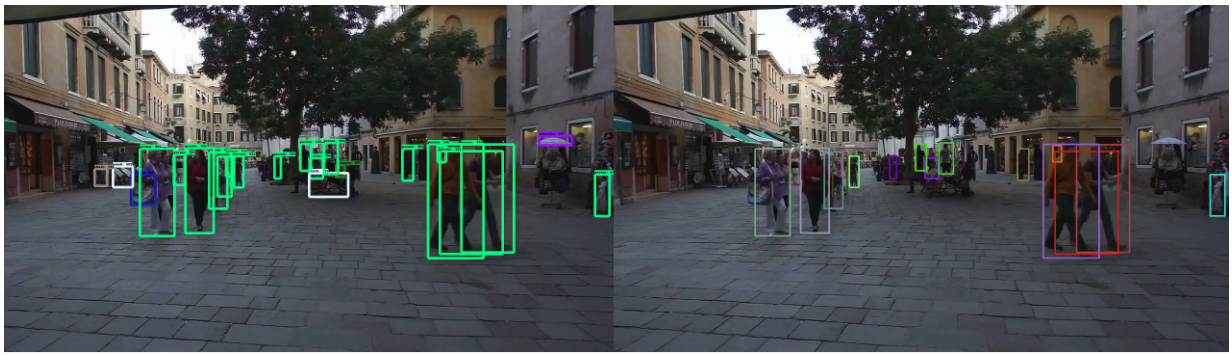


FIGURE 20 – Comparaison annotations fournies par le détecteur (gauche) et bounding boxes générées par le traqueur (droite).

Essais sur la chaîne de traitement témoin avec différents paramètres de SORT

La première tentative d'amélioration des résultats de la chaîne de traitement témoin a été l'étude de l'impact de la modification des paramètres MAXAGE et MINHITS du traqueur SORT sur les résultats de tracking. Le traqueur SORT a été testé selon trois configurations précisées dans le tableau 2 sur les détections du détecteur témoin. Les résultats de ces expérimentations sont fournies en figure 21.

Dans ce tableau, la première ligne de résultats correspond à la configuration témoin du traqueur décrite en partie B/.

Dans la seconde configuration, l'âge maximum d'une trajectoire perdue est fixé à 5 contre 1 dans la configuration témoin. On constate tout d'abord une augmentation du rappel de 5,7% ainsi qu'une augmentation du MOTA de 0,4%. Ces augmentations s'expliquent

Paramètres tracker	Rappel	Precision	F1	MOTA	MOTP	FPS détecteur	FPS tracker
maxage : 1 minhits : 3	46.9	83.7	59.8	36,1	75,7	6,67	101,8
maxage : 5 minhits : 3	52,6	78,7	62,7	36,5	74,9	6,67	87,3
maxage : 5 minhits : 6	50,0	80,8	61,5	36,7	75,2	6,67	93,9

FIGURE 21 – Tableau récapitulatif des résultats de tracking selon les métriques CLEAR avec les trois configurations du traqueur et sur les détections du détecteur témoin.

par le fait que dans cette configuration, le traqueur prédit la position des cibles qui ont été perdues depuis moins de 5 frames. Ce qui permet de fournir des bounding boxes pour les cibles même lorsque celles-ci sont occultées depuis peu. Cela amène une diminution des faux positifs et, dans le cas où l'occultation est très courte, une diminution des échanges d'identifiants. En parallèle, l'augmentation de l'âge maximal des trajectoires perdues provoque l'apparition d'un phénomène de dérive : le filtre de Kalman permettant de prédire la position des annotations finit par se perdre avec le temps, générant ainsi des annotations erronées, se traduisant en des faux positifs, diminuant ainsi la précision et le MOTP. Voir annexe *Résultats complets détaillés des traqueurs selon les métriques CLEAR* pour les valeurs numériques. L'un dans l'autre, la génération de faux positifs et la diminution des faux négatifs et échanges d'identifiants se compensent, provoquant une quasi-stagnation du MOTA. Enfin, on constate une diminution des FPS du traqueur dû à une augmentation du nombre de trajectoires à traiter par la méthode hongroise.

Dans la troisième configuration, l'âge maximum d'une trajectoire perdue est fixé à 5 et L'âge minimal d'une trajectoire à 6 contre 3 dans la configuration témoin. Dans cette configuration, le traqueur ignore des trajectoires qui persistent moins de 6 frames consécutives contre 3 précédemment. On constate globalement une tendance inverse à celle obtenue dans la seconde configuration du traqueur, avec une diminution du rappel et une récupération partielle de la précision, du MOTP et des FPS traqueur, ce qui s'explique par une diminution des faux positifs et des échanges d'identifiants. La valeur du MOTA quant à elle n'augmente que de 0,2% à cause d'une augmentation du nombre de faux négatifs (voir annexe *Résultats complets détaillés des traqueurs selon les métriques CLEAR*).

Analyses préliminaires sur l'influence des paramètres du traqueur

Globalement, on constate que la modification des paramètres du traqueur permet d'influer sur les capacités de filtrage du traqueur sur les détections fournies par le détecteur. L'augmentation de l'âge maximal des trajectoires perdues (MAXAGE) permet de prédire la position de cibles occultées, augmentant le rappel, au prix d'une diminution de la

qualité du positionnement des bounding boxes, diminuant la précision et le MOTP. L'augmentation de l'âge minimal d'une trajectoire (MINHITS) permet quant à elle de filtrer les trajectoires trop fragmentées, et de compenser partiellement le phénomène de dérive dû à la prédiction des bounding boxes. En revanche, la modification de ces paramètres n'a pas d'influence significative sur la consistance du traqueur dans le temps, comme en témoigne la faible augmentation du MOTA.

C/ Nouvelle chaîne de traitement avec entraînement du détecteur

Pour rappel, le détecteur dit "entraîné" est un Faster R-CNN ResNet101 pré-entraîné sur COCO14 et ré-entraîné par transfer learning sur un sous-ensemble des éditions 2017 et 2019 du dataset MOT. Ce sous-ensemble ainsi que les prétraitements qui lui sont appliqués sont précisés en annexe *Chaîne de prétraitement des données MOT*.

Les hyperparamètres de l'entraînement du réseau sont les suivants :

— Backbone : ResNet101
— Optimisateur : Adam
— Poids utilisés pour le transfer learning : Réseau de TFODAPI pré-entraîné sur COCO14
— Nombre de steps avant arrêt : 50000

Par rapport à l'implémentation du détecteur témoin, le nombre maximal de propositions en sortie du Region Proposal Network a été réduit de 300 à 100 afin d'accélérer l'inférence du détecteur.

Résultats détection détecteur entraîné métriques COCO

Le détecteur entraîné a été testé sur les mêmes données que le détecteur témoin et a fourni les résultats suivants :

— Résultat en précision : mAP@0.5 0,83, contre 0,67 pour le détecteur témoin
— Résultat en rappel : mAR@100 0.59 contre 0,51 pour le détecteur témoin

Les détails sur les résultats de l'entraînement et les différentes expérimentations ayant mené à ce résultat sont précisés en annexe *Précisions sur l'entraînement du détecteur*.

Par ailleurs, on constate en visualisant la superposition des bounding boxes sur les images plusieurs améliorations des détections. Tout d'abord, le phénomène de subdivision des bounding boxes lorsqu'un objet est trop proche de l'objectif est en grande partie traité, comme on peut le voir en figure 22. Cela traduit une adaptation du détecteur à cette taille

de cible. Les faux positifs encadrant des vêtements sont moins fréquents. Signifiant une spécialisation du réseau à la détection de piétons. Enfin, les foules ne sont plus détectées comme un individu à part entière, comme on peut le voir en figure 23.

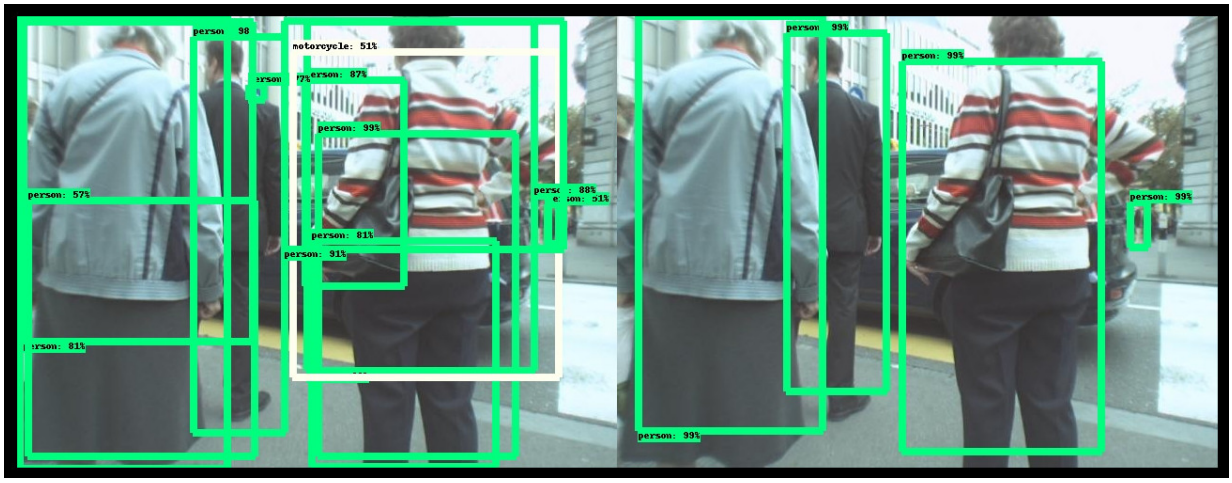


FIGURE 22 – Comparaison d'une détection d'un ensemble de cibles proches de l'objectif par le détecteur témoin (gauche) et par le détecteur entraîné (droite). On constate que le phénomène de subdivision des bounding boxes qui apparaît dans les détections du détecteur témoin n'apparaît pas dans les détections du détecteur entraîné.

FIGURE 23 – Comparaison d'une détection d'une foule par le détecteur témoin (gauche) et par le détecteur entraîné (droite). On constate que le détecteur entraîné ne détecte plus les foules comme un individu unique.

Analyses préliminaires sur l'influence de l'entraînement du détecteur

L'amélioration des résultats de détection peut se résumer en plusieurs adaptations du réseau. Tout d'abord, une adaptation du réseau aux images de la taille de celles du dataset MOT (1920*1080 pixels), une adaptation du détecteur à la taille des cibles par rapport à celle des images et une spécialisation du détecteur à la détection de piétons.

Essais chaîne de traitement entraînée avec différents paramètres de SORT

Comme précédemment, nous avons étudié les effets de la modification des paramètres MAXAGE et MINHITS du traqueur SORT sur la qualité du tracking. Nous avons à nouveau testé le traqueur SORT selon les trois configurations précisées dans le tableau 2 cette fois-ci sur les détections du détecteur entraîné. Les résultats de ces expérimentations sont fournies dans le tableau 24.

Paramètres tracker	Rappel	Precision	F1	MOTA	MOTP	FPS détecteur	FPS tracker
maxage : 1 minhits : 3	48,9	89,1	63,1	41,8	76,0	9,25	129,5
maxage : 5 minhits : 3	55,1	85,6	67,0	45,3	75,2	9,25	111,2
maxage : 5 minhits : 6	52,5	86,9	65,4	43,9	75,4	9,25	104,3

FIGURE 24 – Tableau récapitulatif des résultats de tracking selon les métriques CLEAR avec les trois configurations du traqueur à la suite des détections du détecteur entraîné.

On peut déjà remarquer que toutes les différentes configurations du traqueur ont de meilleurs résultats lorsque couplée au détecteur entraîné que lorsque couplée au détecteur par défaut. Plus précisément, on peut faire plusieurs remarques sur ces résultats :

— En comparant entre les résultats du traqueur dans sa configuration témoin à la suite du détecteur témoin et du détecteur entraîné, on obtient une augmentation de 2% du rappel, de 5,5% de la précision, de 5,7% du MOTA et de 0,3% du MOTP. En regardant les résultats détaillés en annexe *Résultats complets détaillés des traqueurs selon les métriques CLEAR*, on constate que ces résultats s'expliquent par une diminution importante des faux positifs et faux négatifs. Etant donné que dans cette configuration du traqueur, les bounding boxes des cibles perdues ne sont pas prédites par le filtre de Kalman, on peut imputer cette amélioration de suivi au détecteur. Le détecteur est donc plus efficace à reconnaître les piétons. En parallèle, l'augmentation des FPS du détecteur est due à la diminution du nombre maximal de propositions en sortie du Region Proposal Network.

— On remarque que l'évolution des différentes métriques CLEAR est similaire lors de la modification des paramètres du traqueur, lorsque celui-ci est couplé au détecteur par défaut ou au détecteur entraîné. Dans les deux cas, L'augmentation de l'age maximal des trajectoires perdues (MAXAGE) par exemple provoque l'augmentation du rappel, au prix d'une diminution de la précision et du MOTP, pour les raisons précisées en partie B/. En revanche, les effets sur le MOTA, qui étaient négligeables dans le cas du détecteur témoin, sont bien plus notés dans le cas du détecteur entraîné. Cela laisse suggérer que la meilleure qualité de positionnement

des annotations permet de compenser partiellement le phénomène de dérive qui intervient lors de la prédiction des bounding boxes des cibles perdues.

Au vu des résultats selon la métrique F1 et le MOTA, on sélectionne la chaîne constituée du détecteur entraîné et du traqueur dans la seconde configuration comme meilleure chaîne de traitement.

D/ Résultats de la meilleure chaîne de traitement

Afin de se rendre compte de l'importance de l'amélioration du suivi par les modifications successives effectuées sur la chaîne de traitement, le tableau 21 présente les résultats relatifs selon les métriques CLEAR de la meilleure chaîne de traitement face à la chaîne de traitement constituée du détecteur témoin et du traqueur témoin (haut) et de la meilleure chaîne de traitement face à la meilleure chaîne de traitement se reposant sur le détecteur témoin (bas). Le second tableau permettant de rendre compte de l'amélioration de la chaîne de traitement imputable à l'entraînement du détecteur et le premier de l'amélioration globale de la chaîne de traitement toutes modifications confondues.

On retrouve bien les résultats observés lors des conclusions préliminaires précédentes, à savoir que l'amélioration du MOTA et de la précision sont majoritairement dûes à l'entraînement du détecteur, celui-ci détectant moins de faux-positifs, améliorant par la même occasion les FPS du traqueur en limitant le nombre de trajectoires démarrées à partir d'annotations erronées. Et que l'amélioration du rappel est en grande partie imputable à l'augmentation du paramètre MAXAGE, qui permet de diminuer le nombre de faux positifs lors d'occultations de courte durée.

Au final, les résultats les plus intéressants pour notre application sont les nettes augmentations en rappel, MOTA et FPS détecteur, même si le temps d'inférence du détecteur est encore trop élevé pour respecter la contrainte du temps réel.

E/ Conclusion sur l'influence des modifications du réseau

On en conclut que l'entraînement du détecteur sur des données spécifiques permet, par spécialisation de ce dernier sur ces données, d'améliorer d'une part la qualité des détections sur cet ensemble de données et d'autre part de tirer pleinement partie des modifications des paramètres du traqueur. Le détecteur s'habitue à la détection des classes constituant le dataset mais aussi à la définition des images et à la taille des cibles par rapport à la taille de l'image. Cette spécialisation se traduit par une amélioration de la consistance du traqueur et une meilleure gestion des occultations et échanges d'identifiants. Cette amélioration

est d'autant plus notable lorsque la variable MAXAGE est strictement supérieure à 1 car le traqueur est en mesure de prédire la position des cibles temporairement occultées. L'amélioration des détections permet alors de réduire le phénomène de dérive qui apparaît lors de la prédiction de la position des cibles.

Ces analyses sont une première approche à la constitution d'une chaîne de traitement respectant les contraintes données en . De nombreux points restent cependant à améliorer. Ces problèmes ainsi que les pistes d'amélioration éventuelles sont présentées en partie E/.

Par ailleurs, la spécialisation du détecteur doit être réitérée lorsque l'on souhaite traiter des vidéos d'un nouveau genre. Nous aurons par exemple à l'avenir à traiter les données récoltées en collaboration avec Bombardier en juillet dernier (plus de détails sont disponibles en annexe *Document descriptif des objectifs pour la récolte de vidéos sur le labo train de Bombardier*) et qui nécessitera certainement un ré-entraînement du détecteur.

Détecteur	Paramètres tracker	Rappel	Precision	F1	MOTA	MOTP	FPS détecteur	FPS tracker
témoin	max age : 1 min hits : 3	46.9	83.7	59.8	36,1	75,7	6,67	101,8
entraîné	max age: 5 min hits : 3	55,1	85,6	66,9	45,3	75,2	9,25	111,6
Résultats relatifs		+17,1%	+2,3%	+11,9%	+25.4%	-0,7%	+38.6%	+9,2%

Détecteur	Paramètres tracker	Rappel	Precision	F1	MOTA	MOTP	FPS détecteur	FPS tracker
témoin	max age : 5 min hits : 3	52,6	78,7	62,7	36,5	74,9	6,67	87,3
entraîné	max age: 5 min hits : 3	55,1	85,6	66,9	45,3	75,2	9,25	111,6
Résultats relatifs		+4,8%	+8.8%	+6.7%	+24.1%	+0.4%	+38.6%	+27,8%

FIGURE 25 – Tableau récapitulatif des résultats relatifs en terme de suivi selon les métriques CLEAR de la meilleure chaîne de traitement face à la chaîne témoin (haut) et de la meilleure chaîne de traitement face à la meilleure chaîne de traitement se reposant sur le détecteur témoin (bas).

Pistes d'amélioration envisagées

En comparant les résultats obtenus avec la meilleure chaîne de traitement en partie Résultats expérimentaux aux objectifs affichés en partie Qualification du besoin, on peut constater plusieurs défauts de la meilleure chaîne de traitement qui restent encore à corriger.

A/ Problème de temps d'inférence de la chaîne de traitement

Premièrement, le temps d'inférence de la chaîne de traitement est encore trop élevé. Nous visons à créer une chaîne de traitement capable de fonctionner en temps réel. Or, la meilleure chaîne de traitement fonctionne à 9 FPS, vitesse bien inférieure à la vitesse de capture des images du dataset MOT (25-30 FPS). Comme nous l'avons vu lors des résultats en tableau 21 et 24, la lenteur d'inférence de la chaîne est intégralement imputable à la lenteur d'inférence du détecteur qui est le goulet d'étranglement de la chaîne de traitement en terme de vitesse d'inférence.

Or d'après la thèse de S. Murray[47], qui vise à étudier l'impact de la vitesse d'inférence sur la qualité du suivi en utilisant l'algorithme SORT, la consistance du tracking dans le temps est négativement impactée par une diminution de la vitesse d'inférence du détecteur. L'augmentation du temps moyen entre deux frames consécutives implique une augmentation de l'incertitude en positionnement des cibles dans le plan image. Cela amplifie le phénomène de dérive provoqué par l'étape de prédiction du filtre de Kalman, nuisant ainsi au suivi. Dans l'éventualité où un détecteur aurait une vitesse d'inférence inférieure à la vitesse de capture de la caméra, le détecteur devrait ignorer les images lui arrivant de la caméra pendant qu'il effectue sa détection sur une image précédemment recueillie, augmentant ainsi le temps moyen entre deux frames consécutives fournies au traqueur.

Les résultats présentés précédemment ignoraient la vitesse d'inférence du détecteur, l'intégralité des images étant d'abord traitées par le détecteur avant de fournir l'intégralité des annotations au traqueur. Notre détecteur ayant une vitesse d'inférence inférieure à la vitesse de capture de la caméra, la qualité du suivi s'en retrouverait impactée si la chaîne de traitement était mise dans une application temps réel.

Pour augmenter la vitesse du détecteur, plusieurs solutions sont envisageables :

Tout d'abord, remplacer le modèle du détecteur par un modèle plus rapide. On peut par exemple remplacer le détecteur Faster R-CNN par un modèle de détecteur en une seule étape tels qu'une variante de YOLO[50], SSD[43] ou encore RetinaNet[41].
Il est aussi envisageable de prendre un backbone plus léger. Le backbone ResNet101 possédant par exemple plusieurs variantes moins profondes telles que ResNet50.
Enfin, il est possible de réduire la définition des images en entrée du détecteur, les images du dataset MOT étant de grandes dimensions.

Chacune de ces modifications devra être accompagnée de vérifications en termes de précision et rappel du détecteur ainsi formé.

B/ Problème de surapprentissage sur données MOT

Comme il est décrit en annexe *Précisions sur l'entraînement du détecteur*, les résultats d'entraînement du détecteur fournis par TensorBoard laissent apparaître un surapprentissage (aussi appelé "overfit") important sur les données d'entraînement. Ce surapprentissage signifie que le réseau se spécialise plus que de raison dans la détection des cibles présentes dans les données d'entraînement, au détriment de la détection de cibles n'appartenant pas à cet ensemble (voir [14]). Ceci implique que les résultats en détection peuvent être améliorés en réduisant ce surapprentissage, et ainsi améliorer les résultats en tracking, comme vu en partie C/.

Pour cela, plusieurs solutions existent :

Tout d'abord, le surapprentissage peut survenir lors d'un manque de diversité dans les données d'entraînement. Pour y remédier, il est possible soit d'ajouter des données d'entraînement, soit d'accroître la diversité des données d'entraînement en "augmentant les données" (aussi appelé "data augmentation", voir [8]). La data augmentation consiste en un accroissement artificiel de la diversité des données d'entraînement en appliquant un ensemble d'opérations aux images d'entraînement, comme des translations ou rotations d'image, des modifications de couleur...
Le module de data augmentation inclus dans TensorFlow Object Detection API n'a pas porté les résultats escomptés. Je travaille actuellement sur un module de data augmentation qui sera utilisé en interne chez Railenium pour pallier ce problème.

Une autre piste pour expliquer le problème de surapprentissage serait l'existence d'annotations partiellement occultées parmi les données d'entraînement. En effet, les données MOT étant à la base prévues pour du suivi de piétons, les annotations fournies suivent les cibles même lorsque celles-ci sont partiellement ou totalement occultées. Bien que l'étape de

prétraitement précisée en annexe *Chaîne de prétraitement des données MOT* supprime les annotations trop occultées, certaines cibles restent partiellement occultées. l'apprentissage sur de telles données amène le détecteur à se spécialiser sur des cibles qui ne représentent que partiellement des piétons. Dès lors, il suffirait de changer le dataset d'entraînement du détecteur, tout en conservant MOT pour l'évaluation du traqueur.
Fin Juillet, plusieurs datasets de suivi de piéton se sont révélés être disponibles. On trouve tout d'abord le dataset Citypersons[63], [2], une version alternative du dataset Cityscapes [3] réannoté. Ce dataset a été créé à la suite d'un constat similaire à celui fait juste au-dessus : l'occultation des cibles entrave l'entraînement d'un détecteur sur un sous-ensemble des données d'un dataset de suivi de piétons. Pour pallier ce problème, les créateurs du dataset Citypersons ont réannoté le dataset Cityscapes en ajoutant pour chaque cible le masque de segmentation de son contours, la bounding box de sa partie visible et la bounding box de son contours complet. Voir 26.
Un autre dataset de vidéos dédié au suivi de piétons lors de leur montée et descente du train, nommé PAMELA UANDES Dataset[16] est aussi disponible. Enfin, il nous reste encore à exploiter les données récoltées au labo-train de Bombardier courant Juillet.

FIGURE 26 – Exemple d'annotation sur le dataset Citypersons. A partir de l'image d'un piéton (gauche), le masque du piéton est annoté (milieu) puis les bounding boxes de sa partie visible (à droite, en jaune) et de son contours complet (à droite, en vert) sont annotés. Source : [63]

C/ Amélioration de la consistance du traqueur dans le temps

Comme précisé en partie A/, le traqueur SORT relègue la qualité de la détection au détecteur et n'utilise que les coordonnées des bounding boxes pour associer les annotations entre elles à chaque nouvelle frame. Cette stratégie a pour avantage de réduire grandement

le temps d'inférence du traqueur en échange d'un nombre plus important d'échanges d'identifiants entre les trajectoires, notamment lorsque deux cibles ont des trajectoires proches dans le plan image.
Pour corriger cela, deux solutions sont envisageables :

D'après [47], il est possible d'améliorer les résultats de suivi en modifiant la métrique de similarité utilisée par le traqueur. La métrique de similarité est la fonction utilisée pour évaluer la similarité de deux annotations, le résultat de cette fonction appliquée pour chaque couple d'annotations de la frame précédente et de la frame actuelle forme la matrice de poids sur laquelle la méthode hongroise est appliquée (voir annexe *Précisions sur le filtre de Kalman employé pour SORT*). Dans notre implémentation, la métrique de similarité utilisée est l'IOU entre les bounding boxes des cibles, mais d'autres métriques sont possibles, comme le coût linéaire (linear cost) proposé par Sanchez-Matilla[54] ou encore le coût exponentiel (exponential cost) proposé par Yu et al.[62]. D'après les résultats fournis par [47], ces métriques permettent d'améliorer la consistance du traqueur dans le temps, notamment lorsque les FPS sont faibles. Il est aussi envisageable d'utiliser l'apparence des cibles pour améliorer leur identification. C'est dans cette optique qu'a été conçu Deep SORT [60], qui combine un traqueur SORT à une métrique d'association des annotations. Cette métrique d'association, appelée "cosine softmax loss" se base sur l'apparence des cibles pour les reconnaître après une occultation totale. Cette métrique est apprise sur un ensemble de données de réidentification afin d'être la plus discriminante possible entre les cibles.

D/ Pistes pour les autres sous-tâches de la détection de dangers

Comme précisé en partie Qualification du besoin, la détection de situation de dangers aux alentours des portes automatiques ne se limite pas au suivi des usagers lors de leur approche des portes. Il est aussi nécessaire de détecter les chutes des usagers, ou encore lorsque ceux-ci sont bloqués dans les portes. Pour entraîner nos algorithmes, le dataset BOSS[1] est disponible. Ce dataset est un ensemble de vidéos prises par différentes caméras dans un train où les actions des passagers sont annotées.

Conclusion

Au sortir de ce stage de fin d'études, nous avons réalisé une liste des risques autour de la fermeture des portes automatiques du futur train autonome. De ces risques a été constituée un ensemble de sous-tâches que les algorithmes de suivi doivent être mesure de réaliser. Parmi ces sous-tâches, nous nous sommes dans un premier temps intéressé à la mise en place d'algorithmes de suivi de piétons afin de prévenir la fermeture des portes sur des usagers. Un état de l'art des algorithmes de suivi a alors été réalisé et a mené à la constitution d'une structure type pour notre algorithme de suivi. Une première implémentation utilisant le détecteur Faster R-CNN et le traqueur SORT a été réalisée, plusieurs pistes d'amélioration de cette chaîne ont été explorées, notamment l'entraînement du détecteur sur un dataset spécialisé dans le suivi de piétons et la modification des paramètres du traqueur. Ces premières modifications de la chaîne de traitement sont encourageantes car des améliorations significatives de robustesse du détecteur et de consistance du traqueur dans le temps ont été observées. Néanmoins, la solution proposée ne respecte pas encore le cahier des charges mis en place, notamment en terme de temps d'inférence. Plusieurs pistes d'améliorations ont été proposées pour remédier à ces problèmes et constitueront le point de départ d'une thèse à l'IRT Railenium, en collaboration avec l'IFSTTAR pour laquelle j'ai été embauché.

Annexes

Document descriptif des objectifs pour la récolte de vidéos sur le labo train de Bombardier

Descriptif des scénarii d'acquisition lot 13 Accès Voyageurs

Critères de variabilité

Ces critères sont tirés soit de situations types étudiées dans le cadre du tracking de personnes (voir MOT17, CVPR19 ou PETS17), soit d'une réflexion en interne sur le suivi de passagers lors de la montée/descente de train. Ces critères peuvent se diviser de la manière suivante :

- Faire varier la dynamique de la caméra. Cela permet de mieux appréhender le flou de déplacement (motion blur) des usagers lors de leur déplacement.
- Faire varier l'emplacement et l'orientation des caméras afin de pouvoir décider des meilleures configurations de caméras pour la détection et le suivi d'usagers.
- Assurer une variabilité de fonds et de luminosité dans le but de reconstituer le contexte d'une gare à différents moments de la journée.
- Assurer une grande variabilité de situations "standards", c'est-à-dire arrivant couramment à bord de trains, comme la montée/descente sans encombre d'usagers ; des usager retirant certains vêtements à bord du train, s'adossant à un mur ou s'asseyant sur un siège. Les algorithmes développés doivent être capables de distinguer de manière quasi-certaine une montée/descente d'usagers de ces actions courantes.
- Assurer une variabilité importante de situations "complexes", c'est-à-dire pouvant arriver couramment à bord de trains mais étant volontairement conçu pour être plus complexe à analyser. On peut notamment citer le cas d'usagers qui décident au dernier moment de changer de trajectoire, des usagers dans l'impossibilité de monter/descendre du train dû à un nombre trop important d'usagers à bord/en dehors du train ou encore des usagers faisant signe depuis l'extérieur du wagon sans pour autant entrer dans le train... On peut aussi y inclure des difficultés liées à l'environnement, notamment la présence de surfaces réfléchissantes (flaques d'eau...) provoquant des erreurs de suivi ou encore des occultations partielles ou complète de différentes caméras dues à des objets dans le champ de vision ou encore des gouttes d'eau/poussières sur l'objectif. Ces situations permettent de tester les capacités de généralisation des algorithmes développés à des cas non étudiés précédemment et peuvent être utilisées comme "faux positifs".
- Assurer un nombre minimum de situations "particulières" qui, bien que peu courantes dans un train, possèdent une dangerosité suffisamment importante pour être étudiées. On peut par exemple citer la chute d'usagers en montant/descendant du train, des usagers bloquant volontairement le passage à d'autres personnes ou encore des scènes de violence sur les quais ou dans le wagon. Ces situations peuvent aussi

être utilisées pour tester les capacités de généralisation des algorithmes développés à des cas non étudiés précédemment mais sont aussi utiles pour faire de l'étude de situations à risque.
— Assurer une grande variabilité d'usagers afin d'éviter le surentraînement des algorithmes neuronaux sur une catégorie d'usagers. Cela passe par un nombre important d'usagers différents mais aussi par une variété dans leur vêtements (d'été ou d'hiver) ou d'accessoires (chapeaux, sacs...) voire d'objets encombrants (valises, béquilles, fauteuil roulant, déambulateur...) ou d'animaux. Il pourrait aussi être intéressant de faire figurer des personnes habillées en contrôleur ou avec des maillots réfléchissants.
— Varier la densité d'usagers dans les enregistrements afin de tester la robustesse des algorithmes aux occultations interclasse et aux échanges d'identifiant (voir MOT17).

Autres précisions :
— Certaines situations dites "complexes" peuvent être traitées par de la data augmentation en aval des prises de données.
— Une bonne qualité d'image est requise, mais l'entraînement des algorithmes pourrait nécessiter une réduction de celle-ci en prétraitement des images.
— D'une manière générale, les extraits vidéos n'ont pas besoin d'être longs. Des extraits n'excédant pas 30s à 1min à une cadence de 30-60 FPS semblent suffisants.
— Chaque scénario devra être réalisé à plusieurs reprises par différentes personnes afin de garantir un nombre de vidéos par situation suffisamment important pour être divisé en vidéo d'entraînement et de test.
— Il pourrait aussi être intéressant de filmer des échantillons pour juger qualitativement les résultats des algorithmes. Ces échantillons seraient une combinaison de plusieurs scénarios effectués simultanément par plusieurs acteurs afin de présenter les différents scénarios étudiés en une seule vidéo ainsi que les résultats des algorithmes sur ces différentes situations à des personnes extérieures.

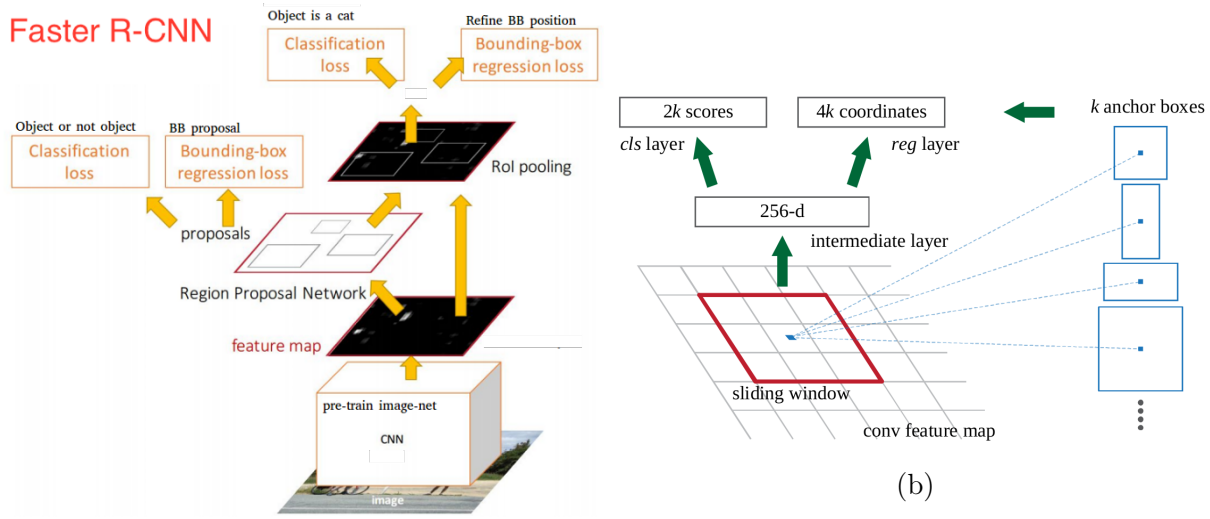
Scénarii possibles

— Vidéo statique ou (faiblement) dynamique.
— Position et orientation des caméras.
  — Position des caméras.
  — Plongée / Contre-plongée.
— Variété des fonds et de la luminosité.
— Types de situations "standards"
  — Ouverture / Fermeture de portes avec descente/montée d'usagers.
  — Ouverture / Fermeture de portes sans descente/montée d'usagers.
  — Scènes d'usagers retirant leur veste ; lisant un livre ; s'asseyant ; s'adossant à un mur...

  - Types de situations “complexes”
    - Ouverture / Fermeture de portes avec descente/montée d'usagers interrompue volontairement.
    - Impossibilité de monter/descendre dû à un nombre trop important d'usagers à bord/en dehors du train.
    - Usagers faisant signe depuis les quais sans entrer dans le train.
    - Mouvements imprévisibles des usagers (voir PETS17).
    - Présence de surfaces réfléchissantes.
    - occultations partielles ou complètes (objets dans le champ de vision ; gouttes d'eau ; poussières).
    - Usagers entrant dans le train par une porte plus lointaine.
  - Types de situations “particulières”
    - Chute en entrée/sortie de wagon.
    - Usagers empêchant l'entrée/la sortie d'autres personnes.
    - Scènes de violence.
  - Types d'usagers
    - Vêtements (d'été ou d'hiver, couleur).
    - Accessoires (chapeaux, sacs...).
    - Objets encombrants (valises, béquilles, fauteuil roulant, déambulateur...).
    - Animaux.
    - Contrôleurs sncf / personnes avec gilet réfléchissant.
  - Densité d'usagers.
  - Cas particuliers
    - Reflets d'usagers sur des surfaces réfléchissantes.
    - Présence d'objets dans l'axe de vision de la caméra provoquant une occultation partielle des usagers sur une ou plusieurs caméras.
    - Scènes de violence sur les quais.

Précisions sur le fonctionnement de Faster R-CNN

Structure détaillée du détecteur Faster R-CNN



(a) (b)

FIGURE 27 – Précisions sur le fonctionnement du détecteur Faster R-CNN. Schéma complet du fonctionnement de Faster R-CNN (gauche) et schéma de présentation du fonctionnement des bounding boxes (droite). Sources : [4], [51]

La structure du détecteur Faster R-CNN[51] se décompose de la manière suivante :

Un réseau neuronal convolutif appelé couramment "backbone" se charge de prétraiter l'image. Il s'agit d'un réseau initialement utilisé pour de la classification d'images détourné comme extracteur de caractéristiques (réseau "CNN" en figure 27a).
A la suite de ce backbone vient se greffer le Region Proposal Network (RPN) qui est un réseau neuronal convolutif chargé de détecter des zones d'intérêt, c'est-à-dire possédant potentiellement une cible. Le RPN fonctionne de la manière suivante :
Sur la dernière couche du backbone est appliquée une fenêtre glissante de dimensions 3*3 qui réduit la dernière couche du backbone en une feature map de dimensions inférieures, voir figure 27b. A chaque application de la fenêtre glissante sont extraites plusieurs régions à partir de k formes de bounding boxes prédéfinies appelées anchor boxes (voir sur la partie droite de la figure 27b).
Les "region proposals" en sortie du RPN consistent en un score d'"objectness" (score de classification binaire entre les classes "objet" et "pas d'objet") issu d'un classificateur et des 4 coordonnées de la bounding box pour chaque anchor box. Enfin, seules les annotations de score supérieur à une valeur seuil sont fournies par le RPN. Au total, le RPN peut au

plus fournir $W * H * k$ zones d'intérêt, où W et H désignent respectivement la largeur et la longueur de la dernière couche du backbone. C'est pourquoi il est assez courant de limiter le nombre de propositions du RPN afin de réduire le temps d'inférence du détecteur.

Donc pour d anchor boxes sélectionnées, les zones d'intérêts en sorties du RPN se présentent sous la forme d'un vecteur de $6 * k * d$ éléments, $2kd$ éléments pour le score d'objectness et $4kd$ éléments pour les dimensions des anchor boxes.

Enfin, chaque zone d'intérêt est détourée de la feature map et est fournie au second étage du détecteur, composé d'un classificateur et d'un réseau de régression dédié à l'affinement des coordonnées des bounding boxes (voir Region of Interest Pooling, RoI Pooling en figure 27a). Le classificateur se charge d'attribuer une classe parmi celles du dataset traité. De plus, dans l'éventualité où plusieurs annotations détecteraient la même cible, on supprime l'annotation de score de classification le plus faible (Non Maximum Suppression, ou NMS).

Il y a donc au final 4 réseaux pouvant être entraînés au sein du détecteur Faster R-CNN : deux réseaux pour le RPN et deux autres pour l'étage secondaire. Le RPN et l'étage secondaire ont chacun leur fonction d'erreur décrite ci-après.

Fonction d'erreur du RPN

Soient t_i le vecteur des 4 coordonnées de l'anchor box d'indice i et p_i sa probabilité de contenir un objet.

Soit p_i^* le label de l'annotation appartenant à la vérité terrain correspondant à la zone de l'image décrite par l'anchor box d'indice i. Ce label peut prendre comme valeur 1 s'il existe un objet à l'emplacement décrit par l'anchor box i, ou 0 dans le cas contraire. Dans le cas où p_i^* vaut 1, soit t_i^* le vecteur de coordonnées de la bounding box de cet objet de la vérité terrain. La fonction d'erreur utilisée pour la rétropropagation du RPN est alors décrite en équation 2 :

$$L(p_i, t_i) = \lambda_{cls} \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda_{reg} \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \tag{2}$$

Où $L_{cls}()$ est la fonction d'erreur en classification (objectness) et $L_{reg}()$ est la fonction d'erreur en régression. N_{cls} est la grandeur du minibatch, N_{reg} est le nombre d'anchor boxes et λ_{cls} et λ_{reg} sont des coefficients de pondération. La fonction $L_{cls}()$ sélectionnée dans notre application est la fonction softmax qui se calcule selon la formule en équation

3. la fonction $L_{reg}()$ utilisée est la "Smooth L1" décrite en équation 4.

$$L_{cls}(p_i, p_i^*) = \frac{e^{p_i - p_i^*}}{e^{p_i - p_i^*} + e^{(1-p_i) - p_i^*}} \tag{3}$$

$$L_{reg}(t_i, t_i^*) = \begin{cases} |t_i - t_i^*| \ si \ |x| > 1 \\ (t_i - t_i^*)^2 sinon \end{cases} \tag{4}$$

La fonction d'erreur utilisée pour le RPN est donc une somme pondérée entre une erreur d'objectness (erreur de classification entre deux classes : "objet" ou "fond") et une erreur de régression sur la position et les dimensions de la bounding box de la zone d'intérêt.

Fonction d'erreur du second étage du détecteur

Pour le second étage du détecteur, l'erreur utilisée est aussi une somme pondérée semblable à celle en équation 2 entre une erreur de classification des zones d'intérêt selon N classes distinctes et une erreur de régression d'affinement de la position des bounding boxes. L'erreur en classification est une fois de plus basé sur une fonction softmax mais cette fois-ci à N éléments au lieu de 2 précédemment. Soit V_{cls} le vecteur de classification de l'étage secondaire et V_{cls}^* le vecteur de classification de la vérité terrain. Chaque élément $V_{cls}^k \in V_{cls} \forall k \in [1, N]$ est le poids estimé par le classificateur que l'objet détecté soit de classe k. Et V_{cls}^* est un vecteur de taille N nul partout sauf en l'indice j, j étant l'indice de la classe véritable de l'objet détecté. Alors la fonction d'erreur en classification est donnée par l'équation 5. La fonction d'erreur en régression est identique à celle utilisée par le RPN.

$$L_{cls}(V_{cls}, V_{cls}^*)_k = \frac{e^{(V_{cls}^k - V_{cls}^{*k})}}{\sum_{t \in [1,N]} e^{(V_{cls}^t - V_{cls}^{*t})}} \ \forall \ t \ \in \ [0, N] \tag{5}$$

$$L_{reg}(t_i, t_i^*) = \begin{cases} |t_i - t_i^*| \ si \ |x| > 1 \\ (t_i - t_i^*)^2 sinon \end{cases} \tag{6}$$

Chaîne de prétraitement des données MOT

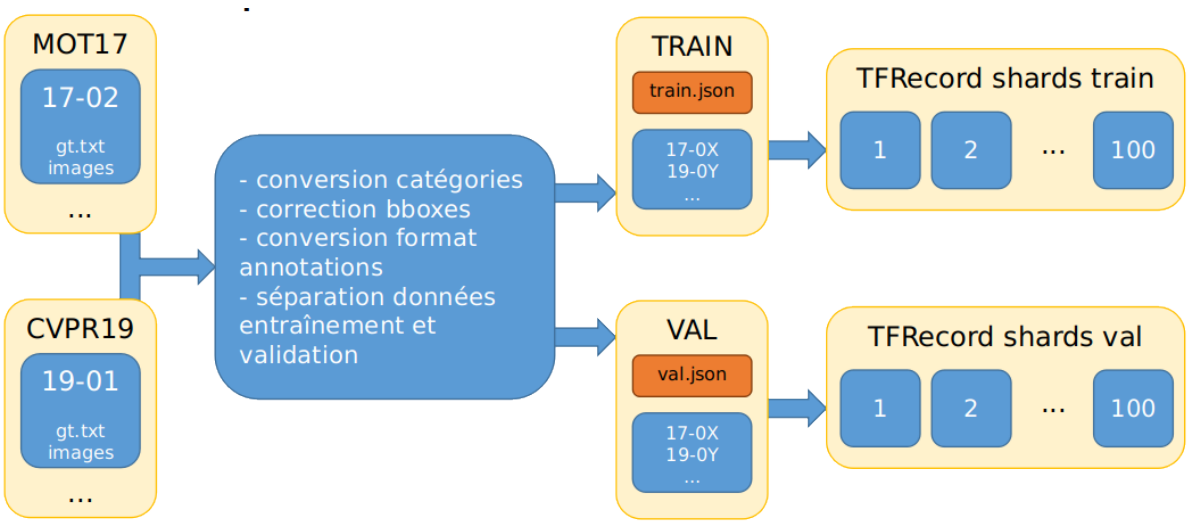


FIGURE 28 – Schéma pipeline de prétraitement des données MOT.

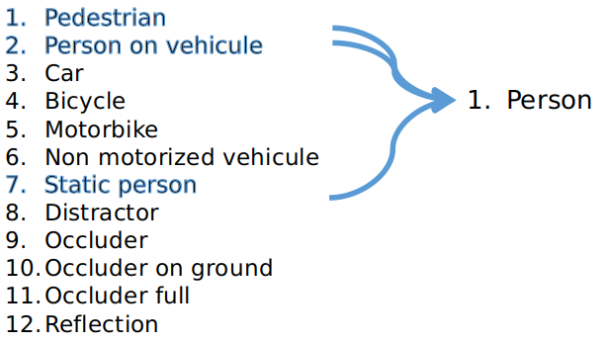


FIGURE 29 – Schéma récapitulatif de la conversion des catégories MOT. A gauche figurent les catégories du dataset MOT et à droite leur équivalent après prétraitement.

La conversion des annotations MOT pour l'entraînement du détecteur se déroule en plusieurs étapes, présentées par la figure 28 et résumées ci-après :

Le dataset MOT est conçu pour le suivi de piétons dans différentes situations de densité de personnes. Les annotations de MOT sont classées selon 12 catégories résumées en figure 29. Bien que plusieurs catégories désignent des piétons (catégories 1, 2, 7), une distinction est effectuée entre personnes statiques, en mouvement ou dans un véhicule. Lors de l'évaluation des résultats de suivi dans le cadre du concours MOT, les annotations désignant des personnes statiques sont ignorées pour ne pas pénaliser des algorithmes de suivi basé sur le mouvement des cibles. De la même manière, les annotations désignant des personnes dans

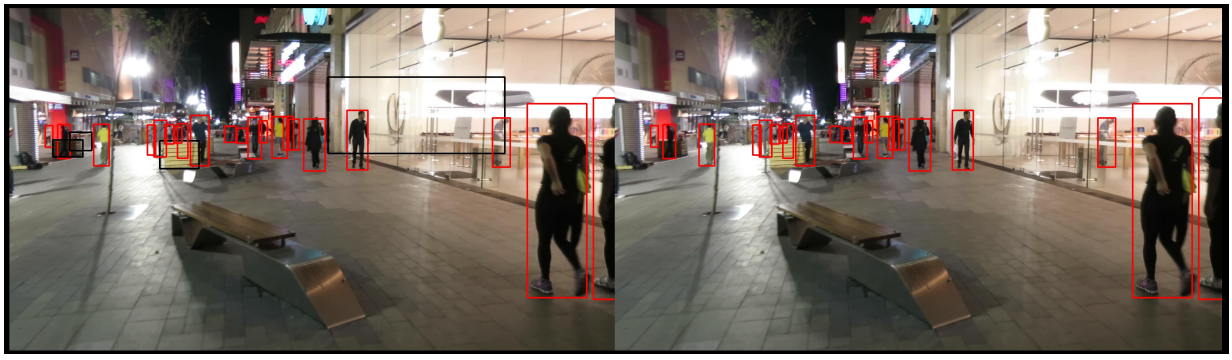


FIGURE 30 – Comparaison entre les annotations MOT originales (gauche) et celles après prétraitement (droite). Les bounding boxes rouges sont celles représentant une personne. Celles en noir ne représentent pas une personne et sont donc supprimées par le prétraitement.

Sous-ensemble	**facile**	**moyen**	**difficile**
Densité moyenne	12,6	55,4	215,0
Ecart-type de densité	4,1	15,7	36,3

TABLE 3 – Tableau de classification des sous-ensembles du dataset MOT selon la densité de personnes à l'écran.

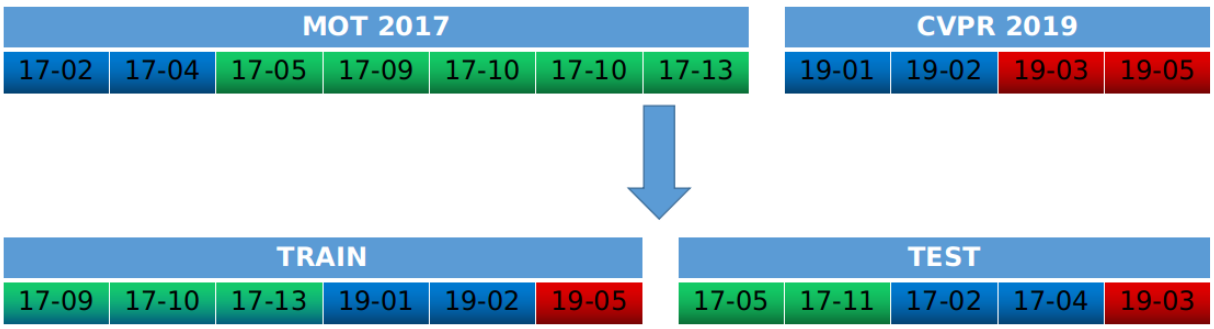


FIGURE 31 – Schéma récapitulatif répartitions données MOT pour entraînement du détecteur. Chaque case symbolise une vidéo du dataset identifiée par son nom de séquence. Les cases vertes, bleues et rouges sont respectivement les vidéos de densité facile, moyenne et difficile

un véhicule sont ignorées pour ne pas pénaliser les algorithmes de suivi basé sur l'identification de silhouettes humaines. Dans notre cas, nous avons besoin d'annoter l'ensemble des piétons présents sur les images, c'est pourquoi les catégories des annotations sont tout d'abord converties selon la table présentée en figure 29.
Dans les annotations du dataset MOT, dans l'optique de conserver un même identifiant

Type	train						test				
ID	17 09	17-13	17-10	19-01	19-02	19-05	17-05	17-11	17-02	17-04	19-03
Densité	10,1	15,5	19,6	61,1	71,8	245,9	8,3	10,5	31	45,3	172,4
Nb frames	525	750	654	429	1391	1657	837	900	600	1050	1202
Proportion	9,7%	13,9%	12,1%	7,9%	25,7%	30,7%	18,2%	19,6%	13,1%	22,9%	26,2%
Mouvement	fixe	mouvant	mouvant	fixe	fixe	fixe	mouvant	mouvant	fixe	fixe	fixe
Angle d'incidence	bas	haut	milieu	haut	haut	haut	milieu	milieu	milieu	haut	haut
Environnement	ext.	ext.	ext.	int.	int.	ext.	ext.	int.	ext.	ext.	ext.
Temps	jour	jour	nuit	jour	jour	nuit	jour	jour	nuages	nuit	nuit

FIGURE 32 – Tableau récapitulatif de répartition des vidéos MOT pour l'entraînement du détecteur. Pour chaque vidéo du dataset MOT sont fournis son identifiant (ID), sa densité moyenne de piétons, son nombre de frames, sa proportion dans le sous-ensemble d'entraînement ou de test, le mouvement éventuel de la caméra, l'angle d'incidence de la caméra, l'environnement (extérieur ou intérieur) et le type de luminosité. Les couleurs des colonnes correspondent à la densité moyenne de personnes, selon le même code couleur qu'en figure 31. Enfin, les chiffres écrits en rouge symbolisent des vidéos dont la moitié des frames ont été supprimées pour équilibrer la proportion de vidéos de densité facile, moyenne et difficile.

pour une cible même lorsque celle-ci est occultée, la bounding box d'une cible encadre l'intégralité de son corps, que celui soit occulté ou non. La bounding box d'une cible peut par ailleurs dépasser du cadre de l'image lorsque celle-ci sort du cadre (voir figure 16). Il devient dès lors nécessaire de supprimer les annotations trop occultées et de limiter les contours des bounding boxes au cadre de l'image afin que le détecteur apprenne sur des données correctes. Heureusement, pour chaque annotation est fourni un indice de visibilité correspondant au taux d'occultation de la cible. Nous avons alors supprimé les annotations dont l'indice de visibilité est inférieur à une valeur seuil et limité les contours des bounding boxes au cadre de l'image.
Les vidéos des éditions 2017 et 2019 de MOT sont alors divisées en données d'entraînement et de validation. Nous avons pour cela classé les vidéos par densité de personnes en 3 catégories : faciles, moyennes et difficiles résumées dans le tableau 3. La répartition des données a été faite afin de garantir une égale représentation des différents critères présentés en figure 32.
Enfin, les données sont converties au format .json, pour pouvoir utiliser la COCOAPI puis au format TFRecord pour être utilisable par TFODAPI.

Précisions sur Tensorflow Object Detection API (TFODAPI)

(a) (b)

FIGURE 33 – Logos de Tensorflow et Tensorflow Object Detection API.

Tensorflow Object Detection API [23] est une librairie Python Open Source spécialisée dans l'implémentation de réseaux neuronaux dédiés à la détection et à la segmentation d'objets. Cette librairie est une extension de la librairie Tensorflow et est encore au stade expérimental. TFODAPI permet d'implémenter des modèles de détecteur et de pouvoir lancer des entraînements ou des inférences à partir de simples fichiers de configuration. TFODAPI est compatible avec Tensorboard[21], une interface graphique pour interpréter les fichiers de log générés par Tensorflow lors d'un entraînement.
TFODAPI procure un ensemble de détecteurs dans son model zoo[22] pré-entraînés et testés sur différents datasets de détection tel que COCO[42]. Pour le moment, seules des variantes de Faster R-CNN, RFCN et SSD sont disponibles avec différents backbones pour les algorithmes de détection et mask-RCNN pour les algorihtmes de segmentation d'instances.
Le format de données utilisé par TFODAPI est le format TFRecords (LIEN), mais de nombreux codes de traduction entre formats de fichiers sont fournis, comme un script de conversion du format COCOAPI vers TFRecords, utilisé dans notre implémentation.
Enfin, TFODAPI supporte nativement plusieurs métriques d'évaluation dont la métrique COCO utilisée dans notre implémentation.

Précisions sur le filtre de Kalman employé pour SORT

L'objectif du traqueur SORT est de suivre une cible au cours du temps. Cette cible est résumée à sa bounding box décrite par 4 paramètres : son centre (u, v), son aire s (8) et son ratio r (9). On définit à partir de ces coordonnées et de leurs dérivées temporelles le vecteur d'état de la bounding box X_k (7) dont on vise à évaluer l'évolution au cours du temps.

Le filtre de Kalman implémenté dans le traqueur SORT se base sur plusieurs équations dont celles présentées équation 10. La première équation, appelée équation d'évolution, permet d'obtenir le vecteur d'état futur X_{k+1} à partir du vecteur d'état actuel X_k, d'un vecteur de contrôle U_k et d'un bruit d'état α_k. La seconde équation, appelée équation d'observation, permet d'obtenir le vecteur d'observation Z_k à partir du vecteur d'état actuel X_k et d'un bruit d'observation β_k.

$$X = [u, v, s, r, u', v', s']^T \tag{7}$$

$$s = w * r \tag{8}$$

$$r = w/h \tag{9}$$

$$\begin{cases} X_{k+1} = F_k X_k + B_k U_k + \alpha_k \\ Z_k = H_k X_k + \beta_k \end{cases} \tag{10}$$

Quelques remarques sur ces équations :

— Tout d'abord, nous ne faisons que suivre les bounding boxes, ce qui signifie que nous n'avons aucun contrôle sur les coordonnées des bounding boxes. Cela implique $U_k = 0$.

— On suppose ici la vitesse de la cible constante dans l'espace image entre deux frames consécutives. Cela explique pourquoi le vecteur d'état X_k ne contient que les dérivées premières des coordonnées des bounding boxes. Par ailleurs, en observant la forme de la matrice F_k, on remarque que l'équation 1 se résume à un schéma d'Euler plus un bruit d'état α_k.

— Les bruits d'état α_k et d'observation β_k sont supposés gaussiens de matrice de covariance respectives Q_k et R_k. On peut remarquer qu'une plus grande incertitude en observation est donnée pour l'aire et le ratio des bounding boxes. Par ailleurs on peut constater dans la matrice d'incertitudes initiales du vecteur d'état P_0 que l'on donne une plus grande incertitude initiale aux dérivées des coordonnées des bounding boxes.

$$F_k = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad Q_k = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10^{-2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10^{-2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 10^{-4} \end{pmatrix}$$

$$B_k = 0 \qquad H_k = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

$$R_k = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 10 \end{pmatrix} \qquad P_0 = \begin{pmatrix} 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10^4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10^4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 10^4 \end{pmatrix}$$

TABLE 4 – Matrices utilisées dans le filtre de Kalman du traqueur

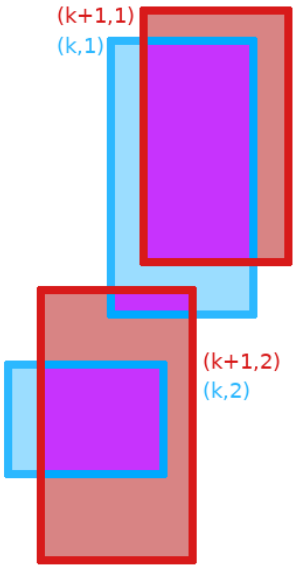
L'ensemble des valeurs de ces matrices sont disponibles en table 4.

Comme décrit en partie A/, l'application du filtre de Kalman se déroule en 4 étapes. On précise ci-après ces 4 étapes ainsi qu'un exemple applicatif sur des données fictives présenté en figure 34.

- La prédiction des coordonnées des bounding boxes de la frame précédente dans la frame actuelle par application de la première équation du système d'équations (10). On génère ainsi les bounding boxes estimées symbolisées en bleu dans la figure 34a.
- L'observation de nouvelles bounding boxes fournies par le détecteur par application de la seconde équation du système d'équations (10). On génère ainsi les bounding boxes observées symbolisées en rouge dans la figure 34a.
- L'association des bounding boxes estimées avec les bounding boxes observées. Pour cela, on calcule l'IOU pour chaque couple [bounding box estimée, bounding box observée]. On obtient ainsi une matrice de poids présentée en figure 34b. On applique alors la méthode hongroise sur cette matrice de poids, qui nous fournit les couples de bounding boxes $[(k,1),(k+1,1)]$ et $[(k,2),(k+1,2)]$. Enfin, on applique un seuil sur les valeurs d'IOU. Si par exemple on ne sélectionne que les annotations d'IOU supérieure au seuil de 0.5, alors seul le couple $[(k,1),(k+1,1)]$ sera sélectionné dans notre exemple en figure 34.

— Enfin, pour chaque couple de bounding boxes [bounding box estimée, bounding box observée], une bounding box corrigée est créée à partir de la formule de correction du filtre de Kalman (non détaillée ici).

Enfin, les bounding boxes estimées non associées sont considérées comme perdues, et les bounding boxes observées non associées forment de nouvelles trajectoires.



(a)

	(k+1,1)	(k+1,2)
(k,1)	0,7	0,1
(k,2)	0	0,3

(b)

FIGURE 34 – A gauche, exemple de deux paires de bounding boxes estimées (bleues) et observées (rouges). Les bounding boxes sont notées sous la forme d'un couple (X,y) où X est l'indice de la frame à laquelle a été prise la bounding box, et y l'indice de la bounding box parmi les bounding boxes de la frame d'indice X. A droite, la matrice de poids générée par calcul de l'IOU entre chaque couple (bounding box estimée, bounding box observée)

Retour sur les métriques COCO

Métriques COCO		
Métriques relatives à la précision		
mAP@0.5	mAP@0.75	mAP@[.5 :.95]
mAP small	mAP medium	mAP large
Métriques relatives au rappel		
AR@1	AR@10	AR@100
AR small	AR medium	AR large

TABLE 5 – tableau récapitulatif des métriques COCO au complet.

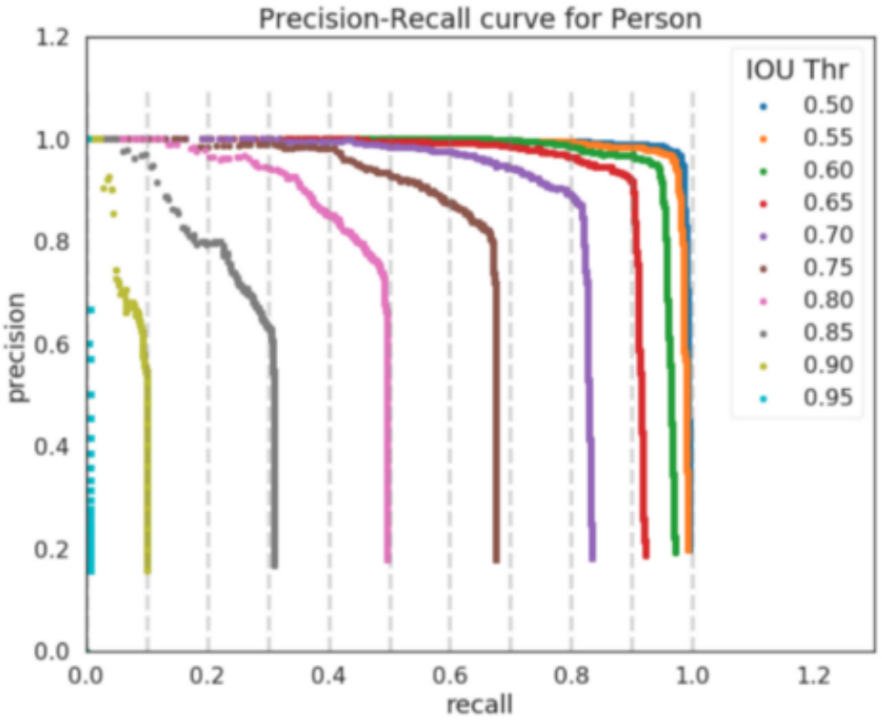


FIGURE 35 – Variation des courbes de précision sur rappel en fonction du seuil d'IOU utilisé. Source : [24]

Les métriques fournies par la COCOAPI sont les métriques utilisées pour l'évaluation de détecteurs dans le contexte du concours Common Objects in COntext (COCO)[42]. Ces métriques se divisent en deux catégories : les métriques relatives à la précision (proportion d'éléments pertinents parmi ceux sélectionnés) et celles relatives au rappel (proportion d'éléments pertinents sélectionnés parmi l'ensemble des éléments pertinents). En partie

B/, nous avions réduit les métriques COCO aux simples précision et rappel au seuil d'IOU de 0.5. Les métriques COCO se déclinent en réalité en 12 métriques distinctes, 6 liées à la notion de précision et 6 autres à la notion de rappel. Ces 12 métriques sont résumées dans le tableau 5.

Métriques COCO relatives à la précision

Les métriques COCO relatives à la précision portent un nom commençant par mAP signifiant "mean Average Precision". Pour comprendre les métriques COCO, il faut s'intéresser au concept de "Average Precision" (AP), et de "mean Average Precision".

Pour calculer l'AP d'un échantillon, on commence par classer l'ensemble des détections réalisées par le détecteur selon leur degré de confiance. En pratique, on utilise le score de classification de l'annotation en sortie du détecteur.
On fixe alors un seuil d'IOU, couramment de 0,5 ou 0,75 et on supprime l'ensemble des annotations dont l'IOU avec la vérité terrain est inférieure à la valeur seuil.
On calcule alors la précision de l'échantillon des annotations pour une valeur de rappel donnée. On réduit progressivement l'ensemble des données étudiées en supprimant les données de degré de confiance les plus faible et on calcule le rappel et la précision sur les données restantes. A chaque évaluation du rappel et de la précision, on vient placer le point (précision,rappel) sur un graphique en deux dimensions. On aboutit après concaténation de tous ces points à la création d'une courbe de précision/rappel comme une de celles présentées en figure 35.
L'AP correspond alors à l'aire sous la courbe précision/rappel. En pratique, l'AP est définie comme la précision moyenne de l'ensemble de valeurs de précision prises à 11 valeurs de rappel différentes espacées équitablement entre 0 et 1. L'ensemble des valeurs de rappel est $Rappel = [0, 0.1, ..., 1.0]$, l'AP est alors définie comme en équation 11.

$$AP = \frac{1}{11} * \sum_{i \in [0,10]} MaxPrecision(Rappel_i) \tag{11}$$

où $MaxPrecision(Rappel_i)$ est la valeur maximale de précision mesurée pour un rappel supérieur à $Rappel_i$.

La mean Average Precision (mAP) est l'AP moyennée pour des seuils d'IOU entre 0,5 et 0,95 par tranche de 0,05 (voir courbes en figure 35). La mAP définie ainsi correspond à la métrique mAP@[.5 :.95] des métriques COCO.
A noter cependant que le terme de mAP est utilisé abusivement pour les autres métriques COCO. En effet, les métriques mAP@0.5 et mAP@0.75 sont en réalité l'average précision aux seuils d'IOU de 0,5 et de 0,75.

La mAP a pour avantage de prendre en compte les scores de classification, la précision des bboxes avec l'utilisation des différents seuils d'IOU et d'être indépendant du nombre d'images ou du nombre d'annotations traitées. Elle est calculée pour une seule classe à la fois, mais on peut aussi calculer sa valeur moyennée sur toutes les classes en cas de détection multiclasses.

Les trois dernières métriques sont relatives à la taille des annotations. Les métriques mAP small, mAP medium et mAP large correspondent à l'average precision pour les annotations dont l'aire en pixels de leur masque de segmentation appartient aux ensembles "small", "medium" et "large" tels que définis dans le tableau 6.

small	medium	large
$Aire < 36^2 pixels$	$36^2 pixels < Aire < 96^2 pixels$	$Aire > 96^2 pixels$

TABLE 6 – Tableau de classification de la taille des annotations en fonction de l'aire de leur masque de segmentation. Source : [42].

Métriques COCO relatives au rappel

D'une manière similaire aux métriques COCO relatives à la précision, les métriques COCO relatives au rappel se résument à 6 métriques décrites dans le tableau 5. L'ensemble de ces métriques se basent sur le principe de l'Average Recall (AR) ou rappel moyen.

Pour calculer l'AR d'un échantillon d'annotations, on commence par classer l'ensemble des détections réalisées par le détecteur selon leur degré de confiance. En pratique, on utilise le score de classification de l'annotation en sortie du détecteur.
On fixe alors un seuil d'IOU, couramment de 0,5 et on supprime l'ensemble des annotations dont l'IOU avec la vérité terrain est inférieure à la valeur seuil.
En calculant le rappel sur au plus les 100 annotations de plus haut score de classification, on obtient la métrique AR@100. En répétant la même opération sur les 10 annotations puis l'annotation de plus haut score de classification, on obtient les métriques AR@10 et AR@1.

Enfin, d'une manière analogue aux métriques mAP small, mAP medium et mAP large, les métriques AR small, AR medium et AR large représentent l'AR calculé sur l'ensemble des annotations dont l'aire du masque de segmentation appartient aux ensembles "small", "medium" et "large" tels que définis dans le tableau 6.

Pour nos analyses, nous avons utilisé les métriques mAP@0.5 pour l'analyse de la précision et AR@100 pour le rappel de nos échantillons.

Précisions sur l'entraînement du détecteur

Le détecteur a été entraîné sur le sous-ensemble des données faciles d'entraînement. Les vidéos MOT ont pour cela subi le prétraitement détaillé en annexe *Chaîne de prétraitement des données MOT*. A noter que le détecteur témoin et le détecteur entraîné ont été testés sur le sous-ensemble des données faciles de test et que ces données n'ont pas été utilisées pour le calcul des résultats du traqueur complet en partie C/.

Mise en évidence du surapprentissage du détecteur

En observant la courbe d'erreur en entraînement, figure 36c, la convergence de la courbe indique que le détecteur a convergé vers une solution. Cette solution fournit une précision de l'ordre de 0,83 mAP@0.5 (voir 36a) et un rappel de l'ordre de 0,59 AR@100 (voir 36b) contre 0,67 mAP@0.5 et 0,51 AR@100 pour le détecteur témoin entraîné sur le dataset COCO14 seul. On constate donc une amélioration du détecteur.
En revanche, la courbe d'erreur en évaluation (figure 36d) indique une augmentation de l'erreur d'évaluation sur le sous ensemble facile des données d'entraînement. Ceci semble indiquer une forme sévère de surapprentissage du détecteur sur les données faciles du sous ensemble d'entraînement. Le surapprentissage, aussi appelé overfit, indique une spécialisation du réseau sur les données d'entraînement (diminution de l'erreur d'entraînement), au détriment de ses capacités de généralisation sur des données n'appartenant pas à son ensemble d'entraînement (stagnation voire augmentation de l'erreur d'évaluation). Le surapprentissage peut être le signe d'un manque de diversité dans les données d'entraînement (le détecteur étant continuellement confronté aux images d'un piéton donné finit par ne plus pouvoir détecteur que ce piéton par exemple) ou à trop grande complexité du réseau par rapport aux données traitées.

Une première tentative de réduction du surapprentissage a été la réduction de la taille du backbone du détecteur. En figure 37 se trouvent les courbes d'entraînement du détecteur Faster R-CNN avec pour backbone un réseau ResNet101 et ResNet50, respectivement de profondeur 101 et 50 couches neuronales convolutives. Ces deux détecteurs ont été entraînés avec l'optimisateur Adam et selon la même configuration : sans augmentation des données ni régularisation ou dropout. Les deux détecteurs ont été entraînés sur le sous ensemble de données faciles et moyennes des données d'entraînement et évalués sur le sous ensemble des données faciles et moyennes des données d'évaluation.

On peut tout d'abord remarquer en figure 37c que le réseau converge dans les deux configurations, avec cependant un temps de convergence plus long pour le détecteur avec ResNet101 en raison de sa plus grande profondeur. Cependant, on constate sur les figures 37a et 37b des résultats en précision et rappel du détecteur nettement inférieurs pour le

modèle basé sur le backbone ResNet50 et une dégradation de ces résultats au cours de l'entraînement. Cette dégradation pouvant être expliquée par la forte augmentation de l'erreur d'évaluation en figure 37d. Ces résultats sont contre-intuitifs par rapport à ce qui est vu régulièrement dans le cas de surapprentissage, et d'autres expériences devront être réalisées avant d'avoir un avis définitif sur la question.

Influence de l'occultation des annotations sur la qualité de la détection

Les courbes en figure 38 visent à évaluer l'influence de la présence d'annotations occultées dans le set de données d'entraînement sur la qualité de l'entraînement du détecteur. Les 4 courbes sont nommées sous la forme "X on Y" où X représente le sous-ensemble d'entraînement et Y le sous-ensemble de test du détecteur. X peut prendre la valeur "easy" (données d'entraînement faciles uniquement) ou "light" (données d'entraînement faciles et moyennes). De même, Y peut prendre la valeur "easy" (données d'évaluation faciles uniquement) ou "light" (données d'évaluation faciles et moyennes). Pour rappel, la classification des vidéos de MOT en données "faciles", "moyennes" ou "difficiles" s'effectue selon la densité moyenne de personnes à l'écran. Les valeurs moyennes de densité pour chaque sous-ensemble de données sont données dans le tableau 3
On peut tout d'abord remarquer en figure 38c que le réseau converge dans toutes les configurations, avec cependant un temps de convergence plus long pour les détecteurs entraînés sur les données faciles et moyennes. L'influence de l'occultation des annotations sur la qualité de la détection est visible en figures 38a et 38b. En effet, les résultats en précision et rappel indiquent une différence de l'ordre de 9% en mAP@0.5 et de 11% en AR@100 entre les détecteurs évalués sur les données "light" et les données "faciles" seules. Cela traduit la difficulté des détecteurs à détecter des piétons dans des foules compactes. En revanche, l'augmentation des données d'entraînement par des données de difficulté moyenne ne semble pas avoir d'impact significatif sur la qualité de la détection, hormis une diminution de l'erreur d'évaluation, qui semble provenir de l'augmentation de la diversité du dataset. Ces résultats semblent indiquer la nécessité d'augmenter la taille du dataset d'entraînement et la difficulté des détecteurs à détecter en présence d'occultation forte.

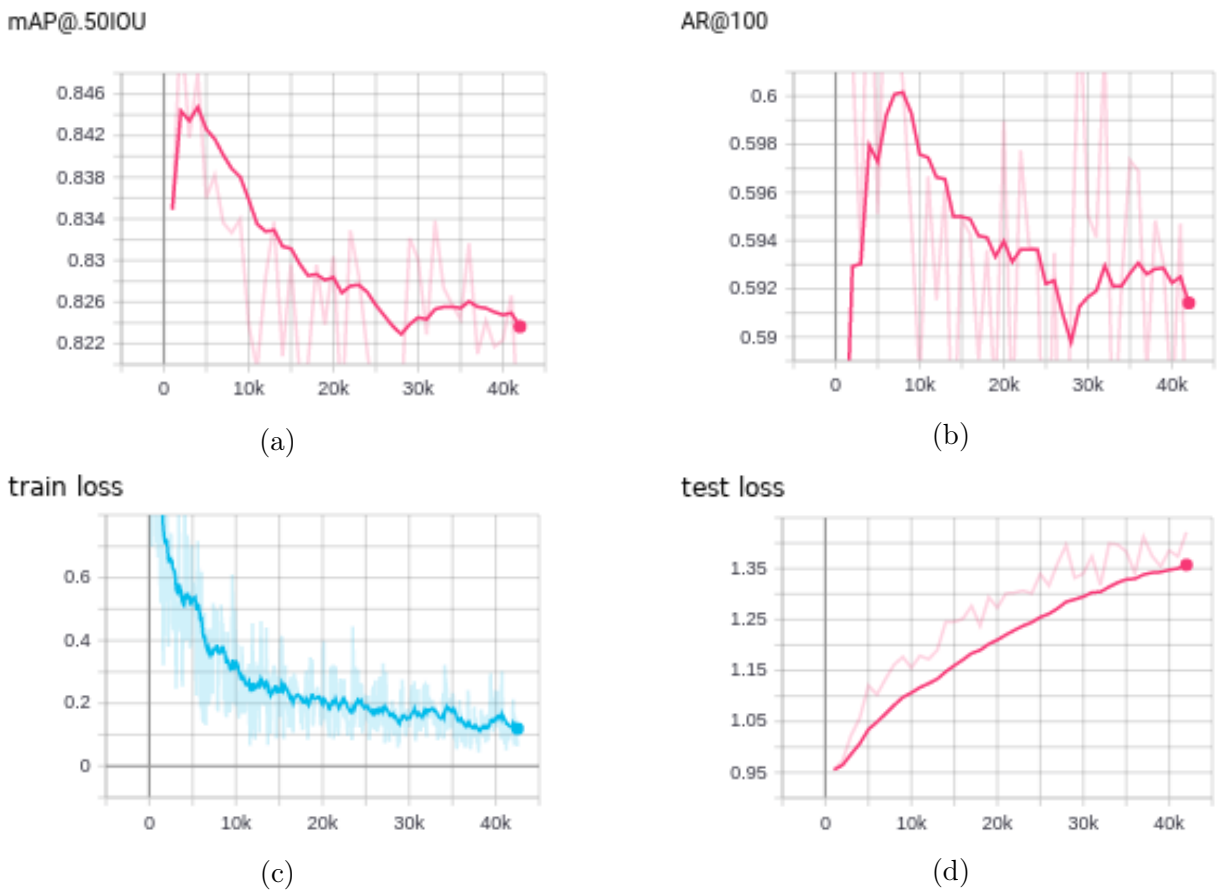


FIGURE 36 – Graphiques des résultats de l'entraînement du détecteur. En première ligne figurent les résultats en précision (mAP@0.5, gauche) et rappel (AR@100, droite) et en seconde ligne les erreurs d'entraînement (gauche) et en évaluation (droite). Tous ces graphiques sont générés sous TensorBoard.

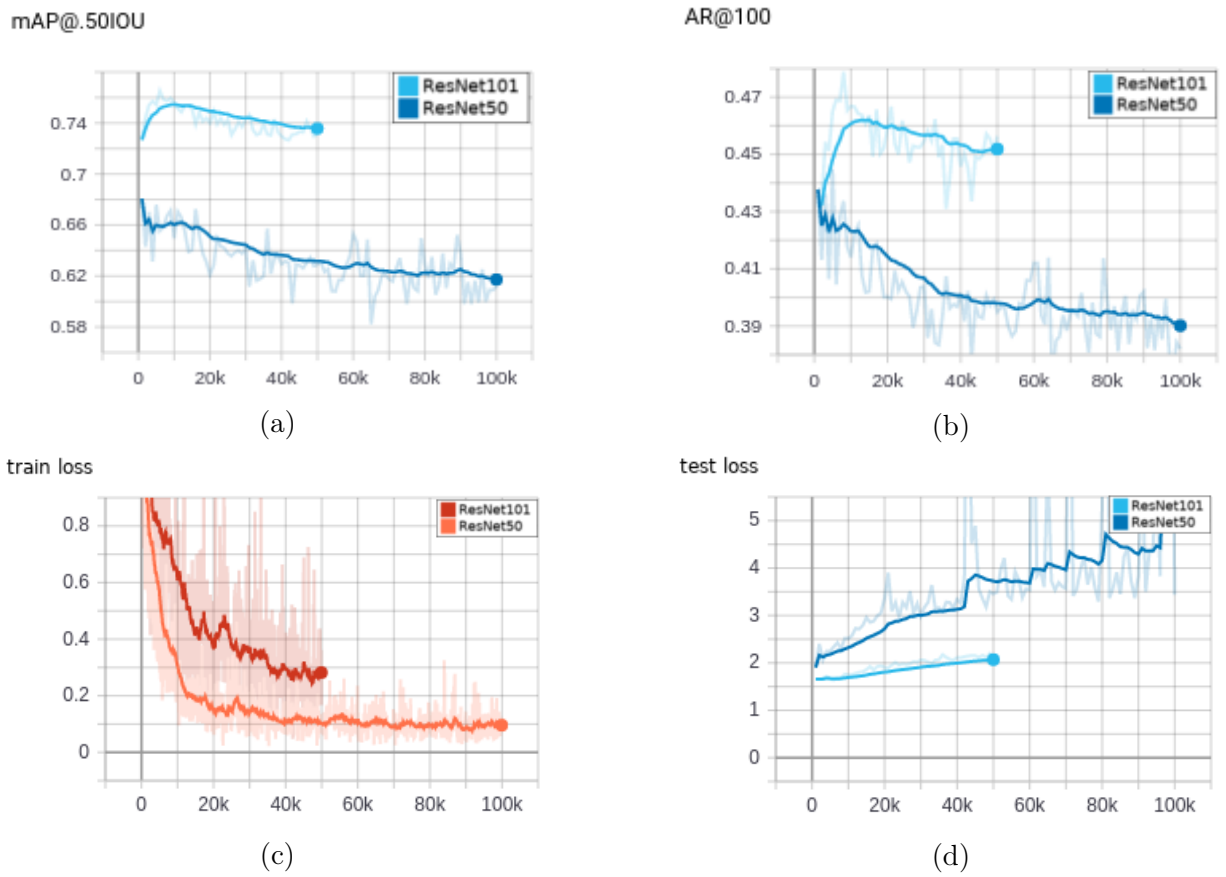


FIGURE 37 – Comparaison des résultats d'entraînement du détecteur avec différents backbones. En première ligne figurent les résultats en précision (mAP@0.5, gauche) et rappel (AR@100, droite) et en seconde ligne les erreurs d'entraînement (gauche) et en évaluation (droite). Pour chaque graphique sont fournis deux courbes correspondant aux entraînements du détecteur Faster R-CNN avec comme backbone un réseau ResNet101 ou ResNet50. Tous ces graphiques sont générés sous TensorBoard.

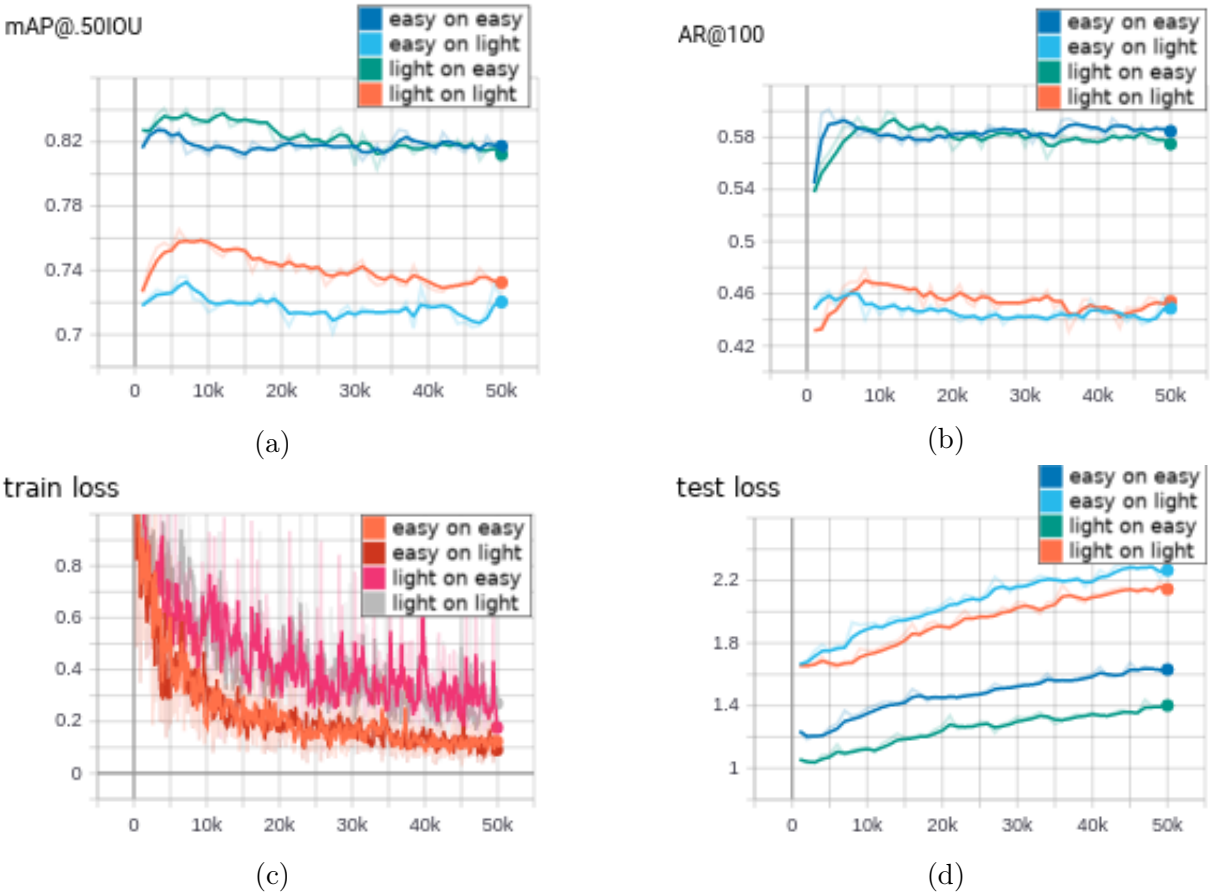


FIGURE 38 – Comparaison des résultats d'entraînement du détecteur sur différents échantillons de données. En première ligne figurent les résultats en précision (mAP@0.5, gauche) et rappel (AR@100, droite) et en seconde ligne les erreurs d'entraînement (gauche) et en évaluation (droite). Tous ces graphiques sont générés sous TensorBoard.

Résultats complets détaillés des traqueurs selon les métriques CLEAR

Modèle Détecteur	Paramètres Trackeur	Rappel	Précision	F1	FP	FN	IDs	MOTA	MOTP	FPS détecteur	FPS tracker
Témoin	maxage : 1 minhits : 3	46.9	83.7	59.8	24675	136449	3182	36,1	75,7	6,67	101,8
Témoin	maxage : 5 minhits : 3	52,6	78,7	62,7	38899	121970	2456	36,5	74,9	6,67	87,3
Témoin	maxage : 5 minhits : 6	50,0	80,8	61,5	32136	128622	2006	36,7	75,2	6,67	93,9
Entraîné	maxage : 1 minhits : 3	48,9	89,1	63,1	15198	131355	3310	41,8	76,0	9,25	129,5
Entraîné	maxage : 5 minhits : 3	55,1	85,6	67,0	23663	115444	2513	45,3	75,2	9,25	111,2
Entraîné	maxage : 5 minhits : 6	52,5	86,9	65,4	19916	122272	1992	43,9	75,4	9,25	104,3

FIGURE 39 – Résultats complets des chaînes de suivi de piétons selon les métriques CLEAR.

Outils internes à Railenium et déroulement du stage

Matériel informatique

Très tôt dans le stage, chaque stagiaire en intelligence artificielle a reçu un ordinateur portable professionnel. Les calculs d'entraînement des réseaux neuronaux étaient effectués sur serveur GPU NVIDIA 4 coeurs (station DGX) disponible aux stagiaires. Nos implémentations se trouvent sur un docker personnel installé sur la station DGX. Le docker utilisé est un docker linux fourni par NVIDIA avec CUDA et CUDNN natifs. La connexion à la station DGX se fait avec JupyterHub et la programmation Python s'effectue sur Jupyter Notebook ou script Python simple. Il est aussi possible de se connecter depuis chez soi aux serveurs de calcul à l'aide d'un VPN interne à Railenium.

Communication avec les encadrants

La communication avec les encadrants s'effectue via plusieurs moyens. Un chat interne (RocketChat) est disponible pour la communication informelle ou les questions de dernière minute, une boîte mail interne à Railenium est utilisée pour les messages plus formels et les demandes de rendez-vous. Chaque stagiaire doit aussi tenir un journal de bord sur le WikiRailenium pour garder une trace de ces avancées et éventuellement les rendre disponibles à d'autres personnes. Enfin, des réunions trimestrielles avec tous les encadrants étaient organisés afin de rendre compte de l'orientation du stage.

Sauvegardes et rigueur de programmation

La sauvegarde des données sur la DGX s'effectue tous les mois mais les données ne sont pour le moment pas enregistrées sur un disque externe. En revanche, les données récoltées sur le labo train ou les datasets téléchargés sont sur un NAS externe sécurisé et j'ai de mon côté effectué régulièrement des sauvegardes de mes codes sur mon ordinateur personnel et sur un disque dur externe.
Les codes développés ont été copiés sur un serveur Gitlab avec un guide d'installation et des descriptifs de code. Concernant les logs lors des entraînements de réseau ou des inférences, j'ai utilisé les logs générés par Tensorflow pour les résultats des réseaux neuronaux et mon propre système de logs pour les autres codes python.

Erreurs de parcours

La structure du code développé en partie B/ résulte de la troisième itération de développement que j'ai réalisée pendant ce stage. Lors de la première itération de développement, mes premiers résultats d'état de l'art en matière de suivi de piétons m'avaient mené vers le développement d'algorithmes de détection de squelettes et de segmentation d'instances comme OpenPose [7] ou Mask R-CNN [34]. Bien que ces solutions semblent très prometteuses pour la détection de postures, elles ne rendent pas compte du mouvement des usagers, ou peuvent nécessiter l'utilisation de réseaux neuronaux plus complexes comme des LSTM. L'utilisation d'algorithmes de suivi multi-objets semblait alors mieux adaptée à notre problème. Cela présentait l'avantage de constituer une méthode de prévention des risques sur les usagers en amont de la situation de danger et leur implémentation était plus simple. L'utilisation d'algorithmes de détection de postures ou d'image captioning trouvera sa place dans la thèse à venir, mais dans un second temps.
Lors de la seconde itération de développement, j'avais trouvé la chaîne de traitement présentée en fin de partie B/. Après des recherches sur les détecteurs basés sur des réseaux neuronaux, mon choix s'était posé sur l'utilisation de YoloV3. Celui-ci promettait un bon compromis entre vitesse d'inférence et efficacité et j'avais trouvé une implémentation en Tensorflow. Malheureusement cette implémentation était peu fiable et mal documentée. Afin de ne pas réitérer cette erreur, j'ai finalement opté pour la librairie Tensorflow Object Detection API qui est testée, possède une documentation exhaustive et une communauté active.

Bibliographie

[1] *BOSS Dataset Website.* disponible sous :
http://velastin.dynu.com/videodatasets/BOSSdata/.

[2] *Citypersons Dataset Bitbucket Repository.* disponible sous :
https://bitbucket.org/shanshanzhang/citypersons/src/default/.

[3] *Cityscapes Dataset Website.* disponible sous :
https://www.cityscapes-dataset.com/.

[4] *Deep Learning for Object Detection, a Comprehensive Review.* disponible sous :
https://towardsdatascience.com/deep-learning-for-object-detection-a-\
comprehensive-review-73930816d8d9.

[5] *Définition de Transfer Learning sur le site Data Analytics Post.* disponible sous :
https://dataanalyticspost.com/Lexique/transfer-learning/.

[6] *Filterpy repository.* disponible sous :
https://github.com/rlabbe/filterpy.

[7] *Openpose Github repository.* disponible sous :
https://github.com/CMU-Perceptual-Computing-Lab/openpose.

[8] *Page datafranca définition augmentation des données.* disponible sous :
https://datafranca.org/wiki/Augmentation_des_données.

[9] *Page de présentation du projet MODSAFE.* disponible sous :
https://www.uitp.org/content/modsafe-0.

[10] *Page Web Concours MOT.* disponible sous :
https://motchallenge.net/.

[11] *Page Web FFMPEG.* disponible sous :
https://ffmpeg.org/.

[12] *Page Web OpenCV.* disponible sous :
https://opencv.org/.

[13] *Page Wikipedia Définition Rappel et Précision.* disponible sous :
https://fr.wikipedia.org/wiki/Précision_et_rappel.

[14] *Page Wikipedia définition surapprentissage.* disponible sous :
https://fr.wikipedia.org/wiki/Surapprentissage.

[15] *Page Wikipedia Méthode Hongroise*. disponible sous : `https://fr.wikipedia.org/wiki/Algorithme_hongrois`.

[16] *PAMELA UANDES Dataset Website*. disponible sous : `http://velastin.dynu.com/PAMELA-UANDES/`.

[17] *Pymotmetrics GitHub Repository*. disponible sous : `https://github.com/cheind/py-motmetrics`.

[18] *Sam Hare research website*. disponible sous : `http://www.samhare.net/research/struck`.

[19] *Simple Online Realtime Tracking Github Repository*. disponible sous : `https://github.com/abewley/sort`.

[20] *Site de l'Union Internationale des Transports Publics*. disponible sous : `https://www.uitp.org/`.

[21] *Site Tensorboard*. disponible sous : `https://www.tensorflow.org/guide/summaries_and_tensorboard`.

[22] *Site Tensorflow object detection model zoo*. disponible sous : `https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md`.

[23] *Tensorflow Object Detection API Github Repository*. disponible sous : `https://github.com/tensorflow/models/tree/master/research/object_detection`.

[24] *Understanding the mAP Evaluation Metric for Object Detection*. disponible sous : `https://medium.com/@timothycarlen/understanding-the-map-evaluation-\metric-for-object-detection-a07fe6962cf3`.

[25] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf : Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[26] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pages 850–865. Springer, 2016.

[27] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468. IEEE, 2016.

[28] Antonio Brunetti, Domenico Buongiorno, Gianpaolo Francesco Trotta, and Vitoantonio Bevilacqua. Computer vision and deep learning techniques for pedestrian detection and tracking : A survey. *Neurocomputing*, 300 :17–33, 2018.

[29] Direction des Statistiques et de La Régulation des Informations Economiques. Sncf open data, trafic de voyageurs et marchandises depuis 1841, 2017.

[30] TU Dresden. First list of hazards, preliminary hazard analysis (pha). Technical report, European Commission, Seventh Framework programme, MODSafe Modular Urban Transport Safety and Security Analysis, 2009.

[31] Sam Hare, Stuart Golodetz, Amir Saffari, Vibhav Vineet, Ming-Ming Cheng, Stephen L Hicks, and Philip HS Torr. Struck : Structured output tracking with kernels. *IEEE transactions on pattern analysis and machine intelligence*, 38(10) :2096–2109, 2015.

[32] Christopher G Harris, Mike Stephens, et al. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

[33] Anfeng He, Chong Luo, Xinmei Tian, and Wenjun Zeng. Towards a better match in siamese network based visual object tracker. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018.

[34] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[36] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. In *European Conference on Computer Vision*, pages 749–765. Springer, 2016.

[37] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[38] Laura Leal-Taixé, Cristian Canton-Ferrer, and Konrad Schindler. Learning by tracking : Siamese cnn for robust target association. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 33–40, 2016.

[39] Laura Leal-Taixé, Anton Milan, Ian Reid, Stefan Roth, and Konrad Schindler. Motchallenge 2015 : Towards a benchmark for multi-target tracking. *arXiv preprint arXiv :1504.01942*, 2015.

[40] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8971–8980, 2018.

[41] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[42] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco : Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[43] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd : Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[44] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2) :91–110, 2004.

[45] Wenhan Luo, Junliang Xing, Anton Milan, Xiaoqin Zhang, Wei Liu, Xiaowei Zhao, and Tae-Kyun Kim. Multiple object tracking : A literature review. *arXiv preprint arXiv :1409.7618*, 2014.

[46] Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. Mot16 : A benchmark for multi-object tracking. *arXiv preprint arXiv :1603.00831*, 2016.

[47] Samuel Murray. Real-time multiple object tracking-a study on the importance of speed. *arXiv preprint arXiv :1709.03572*, 2017.

[48] Luis Patino, Tom Cane, Alain Vallee, and James Ferryman. Pets 2016 : Dataset and challenge. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8, 2016.

[49] IRT Railenium. Livret du nouvel arrivant irt railenium.

[50] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once : Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[51] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn : Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[52] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary R Bradski. Orb : An efficient alternative to sift or surf. In *ICCV*, volume 11, page 2. Citeseer, 2011.

[53] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3) :211–252, 2015.

[54] Ricardo Sanchez-Matilla, Fabio Poiesi, and Andrea Cavallaro. Online multi-target tracking with strong and weak detections. In *European Conference on Computer Vision*, pages 84–99. Springer, 2016.

[55] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv :1409.1556*, 2014.

[56] SNCF. *Des trains autonomes d'ici 2023.* disponible sous : `https://www.sncf.com/fr/groupe/newsroom/trains-autonomes-2023`.

[57] Rainer Stiefelhagen, Keni Bernardin, Rachel Bowers, John Garofolo, Djamel Mostefa, and Padmanabhan Soundararajan. The clear 2006 evaluation. In *International evaluation workshop on classification of events, activities and relationships*, pages 1–44. Springer, 2006.

[58] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[59] Miroslav Trajković and Mark Hedley. Fast corner detection. *Image and vision computing*, 16(2) :75–87, 1998.

[60] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649. IEEE, 2017.

[61] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[62] Fengwei Yu, Wenbo Li, Quanquan Li, Yu Liu, Xiaohua Shi, and Junjie Yan. Poi : Multiple object tracking with high performance detection and appearance feature. In *European Conference on Computer Vision*, pages 36–42. Springer, 2016.

[63] Shanshan Zhang, Rodrigo Benenson, and Bernt Schiele. Citypersons : A diverse dataset for pedestrian detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3213–3221, 2017.

[64] Liang Zheng, Zhi Bie, Yifan Sun, Jingdong Wang, Chi Su, Shengjin Wang, and Qi Tian. Mars : A video benchmark for large-scale person re-identification. In *European Conference on Computer Vision*, pages 868–884. Springer, 2016.

Table des figures

Glossaire

Algorithme glouton	*Algorithme qui suit un choix optimum local*
Anchor box	*Bounding box de taille prédéfinie fixe*
AP	*Average Precision*
AR	*Average Recall*
Augmentation de données	*Accroissement artificiel d'un ensemble de données par application de fonctions aléatoires sur celles-ci*
Backbone	*Réseau neuronal convolutif utilisé pour l'extraction de caractéristiques*
Bounding box	*Cadre englobant au plus près les contours d'un objet*
Caméra fisheye	*Caméra panoramique*
Classification	*Affectation d'un identifiant de classe*
CNN	*Convolutional Neural Network, ou Réseau Neuronal Convolutif*
COCO	*Common Objects in COntext*
Comble-lacune	*Plateforme amovible d'accès au train*
Data Augmentation	*Augmentation de données*
dataset	*Ensemble de données et de leurs annotations*
Deep learning	*Apprentissage profond*
Détection	*Classification et localisation d'un objet*
Detection-based tracking	*Suivi par association*
Détection de postures	*Algorithme de régression de points d'articulation d'objets articulés*
Detection-free tracking	*Suivi par identification*
Feature map	*Vecteur caractéristique de dimension supérieure à 1*
Feature vector	Voir *Vecteur caractéristique*
FN	*False Negative*

FP	*False Positive*
FPS	*Frames Per Second*
Frag	*Fragmentation (d'une trajectoire)*
frame	*Image tirée d'une vidéo*
GoA	*Grade of Automation*
Ground truth	*Vérité terrain*
GT	*Ground Truth*
Hard examples mining	*Extraction d'échantillons non pertinents classifiés à tort comme des échantillons pertinents*
ID	*Identifiant*
IDsw	*Echange d'identifiants*
IFSTTAR	*Institut Français des Sciences et Technologies des Transports, de l'Aménagement et des Réseaux*
Image captioning	*ou sous-titrage d'image, Affectation d'une description textuelle à une image*
IOU	*Intersection Over Union*
IRT	*Institut de Recherche Technologique*
loss	*Erreur*
LSTM	*Long Short Term Memory*
mAP	*mean Average Precision*
MARS	*(dataset), Motion Analysis and Re-identification Set dataset*
Masque de segmentation	*Ensemble des pixels représentant un objet. Souvent simplifié par un polygone*
Méthode hongroise	*ou algorithme de Kuhn-Munkres, algorithme d'optimisation combinatoire qui résout un problème d'affectation en temps polynomial*
Minibatch	*Nombre d'éléments traité par un algorithme avant calcul de l'erreur*
MODSafe	*Modular Urban Transport Safety and Security Analysis*
Monitoring	*Surveillance*
MOT	*Multi Object Tracking*
MOTA	*Multi Object Tracking Accuracy*
MOTP	*Multi Object Tracking Precision*

NMS	*Non Maximum Suppression*
Occultation	*Masquage partiel ou total d'une cible*
Optimisateur	*Méthode de descente de gradient*
PETS	*(dataset), Performance Evaluation of Tracking and Surveillance dataset*
PIA	*Programme des Investissements d'Avenir*
Region proposals	*Régions d'intérêt*
Régression	*Affectation d'une valeur flottante*
Réidentification	*Localisation d'un même objet sur plusieurs vidéos d'une même scène*
Rétropropagation	*(de l'erreur), correction des poids d'un réseau neuronal en fonction de l'erreur effectuée*
Risk assessment	*Identification de dangers*
RoI Pooling	*Region of Interest pooling, ou extraction de régions d'intérêt*
RPN	*Region Proposal Network*
SORT	*Simple Online Reatime Tracking*
Suivi	*Détection et identification d'un objet sur plusieurs frames consécutives*
Surapprentissage	*Spécialisation excessive d'un algorithme sur ses données d'entraînement nuisant à ses capacités de généralisation*
SVM	*Support Vector Machine*
TFODAPI	*Tensorflow Object Detection Application Programming Interface*
Threshold	*Seuil*
Tracker	*Algorithme de suivi*
Tracking	Voir *Suivi*
Transfer Learning	*Apprentissage par transfert, entraînement à partir d'un réseau neuronal pré-entraîné*
UITP	*Union Internationale des Transports Publics*
Vecteur caractéristique	*Vecteur de paramètres résumant un objet*