UCLouvain

École polytechnique de Louvain

Utilisation de descripteurs d'apparence obtenus par réseaux de neurones pour le suivi d'objets par calculs d'affinités dans un graphe

Auteur: **Hugo Fichère**
Promoteur: **Christophe De Vleeschouwer**
Lecteurs: **John Lee, Vladimir Somers**
Année académique 2022–2023
Master [120] : ingénieur civil électricien

Remerciements

A travers ces lignes, je souhaite remercier les personnes qui m'ont soutenu tout au long de ce mémoire.
Tout d'abord, je remercie mon promoteur Christophe De Vleeschouwer pour m'avoir fait découvrir le monde du tracking et m'avoir guidé tout au long de ce projet.
Je remercie ensuite Vladimir Somers et Baptiste Standaert pour leur réactivité à mes nombreux messages et questions.
Je remercie aussi mes amis avec qui cette dernière année à l'université fut fabuleuse.
Pour finir, je remercie ma famille, et plus particulièrement mes parents, qui ont su supporter mes humeurs d'ours tout au long de mes d'études et qui m'ont permis d'en arriver où j'en suis aujourd'hui.

Table des matières

Introduction

Le *Suivi de Multiples Objets*, ou *Multpiple-Object tracking* ou encore *MOT* est utilisé dans de nombreux domaines variés de la vision par ordinateur et traitement d'image :

— La sécurité et vidéo surveillance : le suivi d'objets est largement utilisé dans ce domaine pour observer des personnes, animaux, véhicules ou autres objets suspects dans des zones sensibles. Cela peut aussi servir à détecter des intrusions ou vols.

— Transport et circulation : le suivi de véhicules et de piétons dans des systèmes de gestion de la circulation peut aider à réguler le trafic, à détecter les accidents et à améliorer la gestion des intersections à problème.

— Automatisation industrielle : Dans les environnements de fabrication et de logistique, le MOT peut être utilisé pour surveiller le flux de produits, suivre les mouvements des robots et des véhicules autonomes, optimiser la chaîne d'approvisionnement et améliorer l'efficacité opérationnelle.

— Analyse du comportement humain : Cette technologie peut être utilisée pour suivre et analyser les mouvements des personnes dans des espaces publics ou commerciaux, ce qui peut fournir des informations précieuses pour la planification urbaine, la conception de magasins et l'optimisation de l'expérience client. Cela peut aussi être utilisé dans les études de mouvements de foule.

— Divertissement et réalité augmentée : Dans l'industrie du divertissement, le suivi de multiples objets peut être utilisé pour intégrer des éléments virtuels dans le monde réel, créant ainsi des expériences de réalité augmentée interactives et immersives.

— Recherche scientifique : Les biologistes, les chercheurs en comportement animal et d'autres scientifiques peuvent utiliser le suivi de multiples objets pour étudier les mouvements et les interactions d'animaux ou d'autres objets dans des environnements naturels ou contrôlés. le MOT peut aussi être utilisé en micro-biologie pour étudier le mouvement des cellules.

— Navigation autonome : Dans les véhicules autonomes, le suivi de multiples objets peut aider à détecter et à prévoir les mouvements des autres véhicules, des piétons et des obstacles, ce qui est essentiel pour assurer une conduite sûre et efficace.

— Sport et analyse de mouvement : Le suivi de multiples objets est couramment utilisé dans les sports pour analyser les performances des athlètes, mesurer les trajectoires de balles et de joueurs, et fournir des informations statistiques pour les équipes et les fans.

En somme, le suivi d'objet consiste à suivre et à identifier plusieurs objets individuels à mesure qu'ils se déplacent dans une scène capturée par des caméras ou capteurs. Dans ce mémoire, nous allons nous focaliser exclusivement sur les scènes capturées par une caméra unique.

Plus précisemment, nous allons étudier les utilisations de caractéristiques d'apparence obtenues par un réseau de neurones dans le cadre du suivi d'objets par calculs d'affinités dans un graphe. La question qui nous guidera tout au long de notre processus se formule ainsi : "Est ce que la combinaison du BpbReid avec l'IHT peut permettre d'effectuer correctemment le suivi d'objet, dans des situations où le tracking est reconnu comme difficile, car les descripteurs des cibles sont peu fiables voire parfois inexistants ?"
Le chapitre 1 couvre l'essentiel des notions utiles à la compréhension du sujet du document. Le chapitre 2 visitera brièvement l'état de l'art des techniques actuelles, et détaille le BpbReid et IHT. Le chapitre 3 décris la méthodologie adoptée pour comprendre et implémenter au mieux la fonction de coût entre les noeuds du graphe de l'IHT. Dans le chapitre 4, nous tenterons d'optimiser ces fonctions de coût en cherchant les meilleurs compromis sur les hyper paramètres en fonction des métriques étudiées, avant de conclure.

Chapitre 1

Connaissances de base

Le *multiple object tracking*, ou *suivi d'objets* consiste à détecter et estimer au mieux les trajectoires d'objets qui apparaissent dans une vidéo. Dans chaque image de la vidéo, on peut identifier les objets (des personnes dans notre cas) d'intérêt et tenter d'associer un numéro ou *ID* commun pour chaque détection appartenant au même objet, au cours du temps. Cette tâche est complexifiée par plusieurs facteurs, comme des occlusions, des mouvements de caméra et le mouvement des objets dans l'espace, qui font varier leurs *apparences*. Le tracking peut s'effectuer en temps réel (*online*), ou a posteriori (*offline*). En temps réel, il est plus difficile d'obtenir des résultats corrects, car l'ont ne dispose des images qu'aux temps $t \in \{t_0, t_1, ..., t_i\}$ où t_i correspond au moment associé à l'image de l'instant i. Nous ne disposons donc pas des "futures images" après le temps t_i, qui pourraient mieux nous permettre d'estimer la trajectoire d'un objet. Le désavantage évident de la technique a posteriori est qu'il faut attendre la fin de la vidéo pour commencer le tracking. Dans le cadre de ce mémoire, nous nous limiteront au cas a posteriori.

Figure 1.1 – Processus de tracking complet simplifié

Dans la littérature actuelle, la solution globale fréquemment utilisée, est de

découper le problème en 3 étapes distinctes visibles sur la figure (1.1), qui seront effectuées à la suite :

1. La détection
2. La ré-identification
3. L'association

1.1 Détection

La détection est effectuée par un détecteur de cible, le plus souvent utilisant un modèle de prédiction basé sur un réseau de neurones de type convolutionnel (*CNN*). Il reçoit en entrée une image. Son but est de trouver l'emplacement d'un maximum d'objets réellement présents dans l'image, et d'estimer au mieux leurs emplacements. La détection est alors encadrée par une *délimitation* rectangulaire qui englobe le plus précisemment possible la position de la détection. A sa sortie, nous reçevons pour chaque image un set de positions de détections, et un facteur de confiance associé à chaque détection. Le détecteur `hrnet posetrack18` a été sélectionné pour ce travail.

1.2 ReID

La ré-identification est effectuée par un ré-identificateur dont le but est d'extraire des *caractéristiques* ou *descripteurs* de chaque objet détecté. Ces caractéristiques peuvent correspondre à n'importe quelles caractéristiques de l'objet. Anciennement, il était courant d'utiliser des histogrammes de couleur, formes, etc. Mais depuis que les modèles de réseaux de neurones ont grandement évolué en terme de résultats, il est maintenant plus fréquent et précis d'utiliser des caractéristiques extraites par ces derniers. On utilise le ré-identificateur `BpbReid`[16] dans notre cas.

1.3 Tracking

Une fois le set des détections crée et leurs descripteurs extraits, il ne reste plus qu'à associer dans le temps les différentes détections entre elles, en estimant au mieux leurs trajectoire, sur base des informations que l'on dispose. C'est cette étape qui fait l'objet de l'étude menée par ce mémoire. On représente en général le problème d'association de données [2] via un graphe. Chaque détections correspond à un noeud, et les arêtes reliant les noeuds ont des poids reflétant leurs distances spatio-temporelles et leurs caractéristiques d'apparence. Sur la figure 1.2 la situation est simplifiée sur 3 trois images consécutives. Chaque paires de noeuds qui ne se

situent pas sur la même images est reliée par une arête. Par soucis de clareté, nous omettons les arêtes reliant les noeuds en t_{i-1} à ceux en t_{i+1}

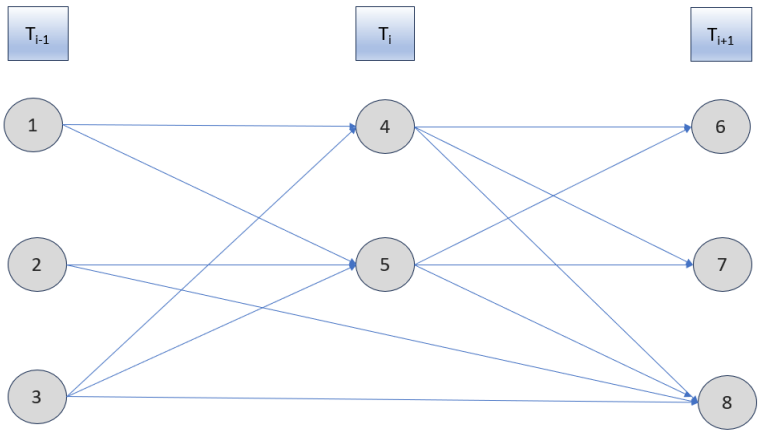


FIGURE 1.2 – Graphe des détections

le problème d'association de données revient à partitionner le graphe en sets disjoints de trajectoire T_i $(i > 0)$ contenant des noeuds, tel que :

- Chaque set contient au maximum une détection à chaque instant. Une détection ne peut être présente qu'une fois maximum sur une image
- Chaque détection n'est contenue que dans un seul set T_i
- Chaque détection de la trajectoire est cohérente avec le reste des autres détections de la trajectoire

Mathématiquement, on peut le formuler de cette manière :

$$\arg\min_{C} \sum_{i=1}^{K} C(T_i) \tag{1.1}$$

$$\text{Sujet à :}$$

$$T_i \cap T_j = \emptyset,$$

$$\cup_{i=1}^{K} T_i = \mathcal{V},$$

$$\forall i > 0, \forall u, v \in T_i, t_u \neq t_v$$

$$\forall i > 0, \forall t, \exists u \in T_i, \text{ tel que } t_u = t$$

où :

- $\mathcal{V}$ représente le set de tous les noeuds du graphe

— $C(T_i) = \sum_{u,v \in T_i, u \neq v} w_{uv}$ représente le coût total de la trajectoire T_i. Il est calculé en additionnant les poids w_{uv} de toutes les arêtes qu'il contient. Ces poids augmentent avec les distance spatio-temporelle et d'apparence entre les deux noeuds.

— t_u représente le temps lié au noeud u

— $\{T_i\}_{i=1}^{K}$ représente le set de toutes les trajectoires

Cependant, la résolution du problème d'association, avec $C(T_i)$ tel que défini plus haut, est NP-complet. En effet, si la trajectoire T_i contient n noeuds, alors le nombre d'arêtes qu'il y a entre tous les noeuds de la trajectoire est $\frac{n(n-1)}{2}$. Nous verrons dans la section 2.2 les simplifications et astuces qui ont été trouvées pour pallier à ce problème.

1.4 Métriques

Evaluer les performances d'un tracker n'est pas une chose aisée. Souvent, elles nécessitent des décisions humaines pour les déterminer. Par exemple, deux détections correspondent entre elles lorsqu'elles se recouvrent le plus possible. A partir du moment où leur score de similarité (d'un point de vue spatial) dépasse un certain seuil, elles sont considérées comme correspondantes. Mais le choix de ce seuil reste humain.

D'autre part, le processus fonctionne en cascade : si la détection est mauvaise, le ReID et le tracker vont eux aussi sortir des résultats incorrects, alors qu'ils peuvent à la base fonctionner correctement.

S'ajoute à cela la difficulté à quantifier les performances de l'association le plus synthétiquement possible. Nous allons voir qu'il existe une multitude de métriques pour l'évaluer, toutes se penchant en particulier plus sur des *cas d'échecs* ou au contraire, des *cas de réussites*, liés à l'association.

Nous allons développer les métriques utilisées dans le `MOTchallenge` [5]. Pour définir les métriques, il faut avant cela examiner les potentiels problèmes qui peuvent survenir lors de l'association.

Avant cela, définissons quelques notations, où *ground truth* signifie *vérité de référence*, et correspond aux informations précises et conforme à la réalité :

— gtDet : détection du ground truth

— prDet : détection prédite par le détecteur

— gtTraj : trajectoire du ground truth, composée de gtDet

— prTraj : trajectoire prédite par le tracker, composée de prDet

— gtID : ID associé à une trajectoire du ground truth

— prID : ID associé à une trajectoire prédite

1.4.1 Cas de figures

Lors du processus de suivi d'objets, trois types d'erreurs peuvent survenir :

1. Erreur de détection : le tracker prédit une détection prDet qui n'est pas dans le ground truth, ou bien échoue à prédire une détection qui fait partie du ground truth
2. Erreur de localisation : le détecteur prédit une détection prDet, mais celle-ci n'est pas alignée parfaitement avec la détection du ground truth gtDet
3. Erreur d'association : le tracker assigne le même prID à deux prDet qui ont une gtID différente, ou bien il assigne deux prID différentes à deux prDet qui ont la même gtID.

Les deux premières erreurs sont indépendantes de l'algorithme d'assignation de prID et résultent d'un manque de précision du détecteur. Leurs conséquences est la création de Faux Positifs (FP) et Faux Négatifs (FN), qui sont expliqués plus bas. La dernière erreur provoque des switch d'ID (IDSW).

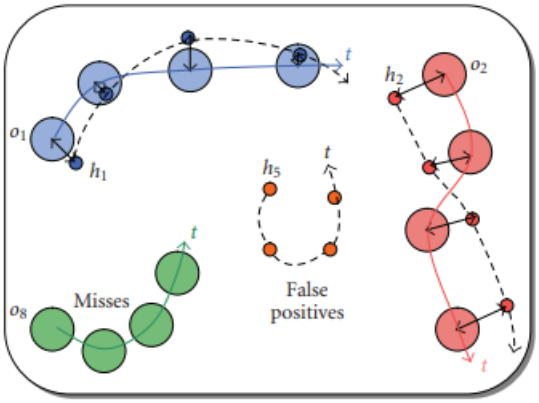


FIGURE 1.3 – Exemple simplifié : h correspond à une hypothèse (prDet) et o un objet (gtDet). En bleu, une situation idéale où la trajectoire est correctement reconstruite. En vert, les gtDet n'ont aucun prDet avec lequel s'associer : FN. En rouge au milieu, les prDet n'ont aucun gtDet avec lequel s'associer : FP. En rouge à droite : une trajectoire est correctement construite mais les prDet sont trop éloignées spatialement des gtDet, et en résulte FN et FP. [4]

1.4.2 Définition des métriques

L'évaluateur de `MOTchallenge` [4] possède un set de métriques appartenant chacune à différentes philosophies : `CLEAR-MOT` [4] qui évalue les métriques images par images, `IDF1` [15] qui se focalise sur les trajectoires entières, et un set de métriques HOTA [11] qui tente de mettre autant d'importance entre la précision de détection et d'association.

Pour chaque métrique, la procédure est la même : une correspondance bijective est effectuée entre les prédictions et le ground truth. Pour chaque paire prédiction-ground truth, un score de similarité est calculé (souvent effectué grâce à IoU [14], *intersection over union*). L'algorithme hongrois est ensuite utilisé pour maximiser le score de correspondance (maximiser la somme des similarités totale). Une fois la correspondance trouvée, toute prédiction associée à un ground truth est appelée TP. Toute prédiction non associée avec un ground truth est appelée FP. Toute détection non associée avec une prédiction est appelée FN.

Clear-MOT

MOTA (Multiple Object Tracking Accuracy) est la seule métrique de `Clear-MOT` qui nous intéresse. C'est la métrique la plus utilisée dans la littérature. Malgré cette popularité, MOTA est fort critiquée car elle donne bien plus d'importance à l'erreur de localisation. Dans `Clear-MOT`, les prédictions et ground truth comparés sont prDet et gtDet, image par image. La définition de MOTA s'écrit comme :

$$MOTA = 1 - \frac{\sum_t FN_t + FP_t + IDSW_t}{\sum_t GT_t} \tag{1.2}$$

Où FN_t représente les faux négatifs sur l'image correspondant au temps t, FP_t les faux positifs sur l'image correspondant au temps t. $IDSW_t$ est un TP au temps t dont le prID diffère du prID de ce TP au temps $t-1$. La valeur de MOTA se situe entre $-\infty$ et 1. Il est intéressant de s'intéresser aux sous métriques de MOTA. Lors de l'évaluation, nous n'allons pas regarder les valeurs de MOTA en tant que telle, mais plutôt à FN, FP et $IDSW$, car ces trois métriques nous donnes des indications différentes.

IDF1

Les métriques `IDF1` mesurent la capacité de préservation d'un ID sur une séquence entière. Dans le paradigme de IDF1, les prédictions et ground truth comparées ne sont plus prDet et gtDet comme dans `Clear-MOT`, mais plutôt prTraj et gtTraj. Les prTraj associées aux gtTraj sont appelées IDTP. Les prTraj non associées à une gtTraj sont les IDFP et les gtTraj non associées à une prTraj sont

les IDFN.
On définit ID-Recall comme :

$$ID - Recall = \frac{|IDTP|}{|IDTP| + |IDFN|} \tag{1.3}$$

ID-Precision :

$$ID - Precision = \frac{|IDTP|}{|IDTP| + |IDFP|} \tag{1.4}$$

Pour finir, la métrique qui prend les deux dernières métriques en compte, pondérées de la même manière :

$$IDF1 = \frac{|IDTP|}{|IDTP| + 0.5|IDFP| + 0.5|IDFN|} \tag{1.5}$$

HOTA

Le paradigme HOTA est la métrique la plus connue sortie récemment. Elle combine un grand nombre de de métrique qui se complètent. Etant assez compliquée et à mi chemin entre MOTA et IDF1, j'ai décidé de ne pas l'inclure dans les métriques à étudier. Cependant, une sous métrique d'HOTA est très intéressante : *Association Accuracy* ou *AssA*. AssA ne se focalise que sur la manière dont un tracker associe correctement le bon prID le long d'une prTraj qui lui est associée.

Comme pour `IDF1`, nous effectuons une correspondance bilatérale entre les prTraj et les gtTraj. Mais cette fois-ci, nous nous concentrons que sur les TP. Pour une trajectoire prTraj donnée, comprise dans le set des TP, on séléctionne $Traj_c$. De manière interne à cette trajectoire, qui est un TP, on définit de nouvelles valeurs :

- TPA(c), le set de détections prDet et gtDet dont le même ID a été associé. Donc le set tel que : prID(prDet) = gtID(gtDet). Ces détections correspondent aux détections correctement associées dans $Traj_c$
- FNA(c), le set des détections appartenant à la trajectoire gtTtraj qui sont mises en correspondance avec $Traj_c$, mais qui ne sont pas comprises dans $Traj_c$
- FPA(c), le set des détections appartenant à $Ttraj_c$ qui sont mises en correspondance avec une gtTraj, mais qui ne sont pas comprises dans gtTraj

Grâce à cela, on peut définir $AssA_{Jaccuard}$ comme :

$$AssA_{Jaccuard}(c) = \frac{|TPA(c)|}{|TPA(c)| + |FNA(c)| + |FPA(c)|} \tag{1.6}$$

Et donc, AssA qui s'applique au set entier de trajectoires, et qui est la moyenne des $AssA_{Jaccuard}(c)$:

$$AssA = \frac{1}{|TP|} \sum_{c \in |TP|} \frac{|TPA(c)|}{|TPA(c)| + |FNA(c)| + |FPA(c)|} \tag{1.7}$$

La figure1.4 illustre ces dernières notions un peu plus subtiles que les précédentes.

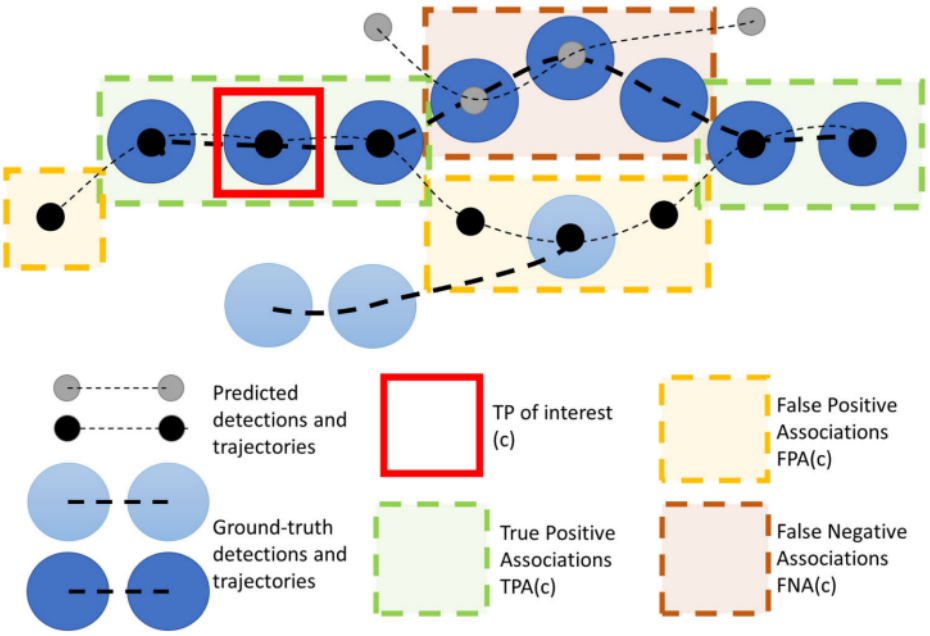


Figure 1.4 – Visualisation de TP, TPA, FPA et FNA

Chapitre 2

BpbReid et IHT

2.1 BpbReid

Comme nous allons le voir dans la section 2.2, l'algorithme qui effectue la tâche d'associer les détections d'une séquence pour former des trajectoire est conçu pour manipuler des caractéristiques sporadiques et avec un degré variable de fiabilité. Une manière d'extraire ces caractéristiques se nomme *ré-identification de personne* et est souvent abbrégé en ReID. Le ReID [16], [19], [13] consiste à retrouver une image de personne, *la requête*, dans une base de données d'une multitude de personnes. Cette méthode est fréquemment utilisée dans le cas de vidéo surveillance. Les caractéristiques de la requête sont extraites par le ReID et ensuite comparées avec chaque caractéristiques des images de la base de données via une fonction de coût (qu'on peut aussi appeler distance). Si la requête possède une distance inférieure à un certain seuil avec une autre identité de la gallerie, alors il y a correspondance entre la requête et cette identité. La figure 2.1 propose un exemple d'une telle situation.

La fonction de coût a été minimisée au préalable lors de l'entrainement, ainsi le modèle qui extrait les caractéristiques des images possède déjà des poids correctes. Dans notre cas, nous n'allons pas chercher une personne dans une base de données, mais plutôt comparer des personnes entre elles via ce ré-identificateur. Toutes les détections passent dans le ReID et en sont extraites les caractéristiques leurs correspondant. Grâce à cela, les détections similaires ont des caractéristiques proches, et vice versa pour les détections qui ne se ressemblent pas.

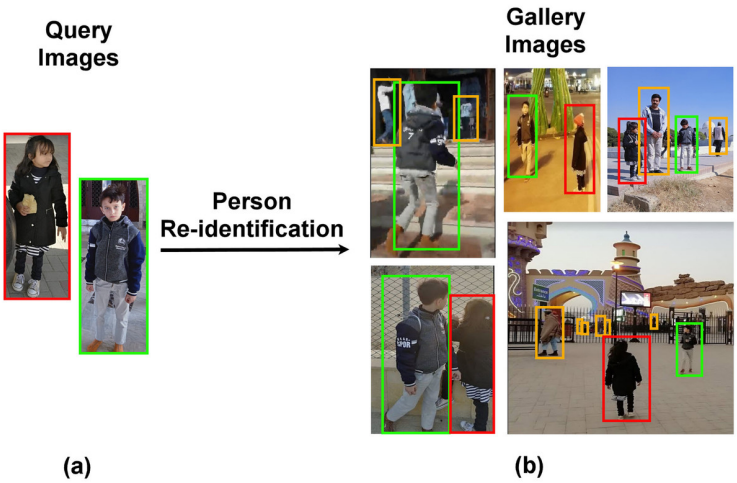


Figure 2.1 – Ré-identification

Dans le cas des ré-identificateurs classiques, deux problèmes majeurs se posent : (1) les délimitations peuvent contenir des détections partiellement cachées par des obstacles ou autres personnes, ce qui induit des fausses informations sur la détection. (2) Lorsque l'une ou deux détections sont partiellement cachées, comparer l'entierté des deux délimitations entières n'est plus pertinent. Par exemple, nous ne pouvons pas comparer simplement un bas de corps d'une personne A avec le haut du corps d'une personne B. La comparaison n'a de sens que lorsque les deux détections sont totalement visibles.

Pour contrer ces problèmes, un nouveau type de ré-indentificateur a été conçu : le ReID basé sur les membres du corps [17],[21],[22]. Un nouveau problème en découle : ces ré-identificateurs utilisent la même fonction de coût que pour les méthodes globales. Malheureusement, ces fonctions de coût fonctionnent avec le principe que différentes identités ont forcément des apparences différentes. Le ReID utilisé dans le cadre de ce mémoire, BpbReID, résout ce problème en fonctionnant avec une nouvelle fonction de coût, spécialement conçue pour le body part : `GiLt`. Son fonctionnement sera développé plus tard.

Pour finir, un dernier problème persiste : comme l'entrainement du modèle de ReID est supervisé, il nécessite une base de données où les différentes parties du corps sont annotées, humainement. Bpbreid utilise une astuce élégante pour contourner ce problème en ne nécessitant un nombre d'annotation que pour l'entrainement des poids de son réseau de neuronnes. Il n'y a pas besoin d'avoir des images dont les membres du corps sont annotées pour utiliser BpbReid par la suite.

2.1.1 Fonctionnement du BpbReID

La figure 2.2 indique le processus du BpbReid. La partie supérieure concerne l'entrainement du réseau de neuronnes, et la partie inférieure montre comment fonctionne la ré-identification entre deux détections différentes. L'explication se découpe en quatre modules.

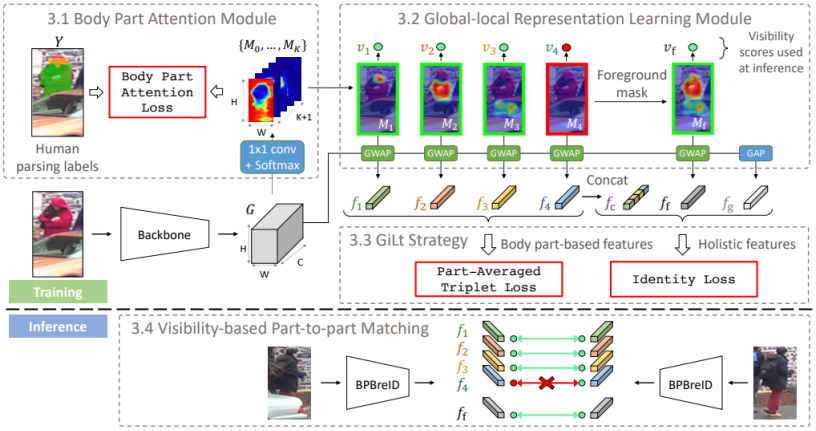


FIGURE 2.2 – Processus entier du BpbReID [16]

Extraction des caractéristiques

Au début de la chaine du processus, une image (qui correspond à un tenseur de taille $R^{3\times H\times W}$, car l'image est en couleur), est fournie en entrée. Le tenseur passe dans un squelette de réseau de neuronnes classique [10], qui en extrait une carte d'apparence, G, de taille $R^{H\times W\times C}$. On peut voir G comme un gros tenseur de caractéirstiques pas encore expoitables.

Classificateur pixel par pixel

La prochaine couche est une convolution combinée d'un softmax [8]. La convolution 1x1 permet d'extraire pixel par pixel de G, et le softmax attribue un vecteur de probabilité de taille $R^{1\times(K+1)}$ à ce pixel, dont chaque entrée correspond à la probabilité du pixel d'appartenir au k-ième membre du corps, et dont la somme totale est unitaire. La première entrée correspond à la classe *arrière plan*. Le tenseur finalement obtenu est donc de taille $R^{H\times W\times(K+1)}$. Il est appelé score de classification. A chaque pixel de l'image, est associée un vecteur de probabilité d'appartenance à une partie du corps, qui peut être vu comme une collection de $K+1$ cartes de probabilité M_k. On a donc classifié les pixels de l'images en fonction de leur probabilité d'appartenir à un certain membre du corps.

Pour que l'entrainement soit supervisé, et donc que les paramètres du réseau de neuronnes puissent être correctement déterminés, il faut un ground truth auquel on compare nos M_k, via une fonction de coût. Ces ground truth sont obtenus sur des images Y, en segmentant à la main les différentes parties du corps de la personne présente sur la détection. Pour chaque vecteur de probabilité d'appartenance à un membre du corps du ground truth, les pixels correspondant à cette partie du corps sont initilisés à 1, et 0 pour les autres.

La fonction de coût utilisée pour comparer les M_k et Y_k est formulée comme :

$$L_{pa} = -\sum_{k=0}^{K}\sum_{h=0}^{H-1}\sum_{w=0}^{W-1} q_k log(M_k(h,w)) \tag{2.1}$$

$$\text{avec } q_k \ = \begin{cases} 1 - \frac{N-1}{N}\epsilon, & \text{si } Y(h,w) = k. \\ \frac{\epsilon}{N}, & \text{sinon.} \end{cases} \tag{2.2}$$

Où N est la taille du batch, ϵ est un paramètre qui représente le taux d'adoucissement de régularisation [16].

L'avantage de ce module est double : il fournit au prochain module les K cartes de probabilité, et il permet en même temps au modèle d'affiner ses M_k qui seront plus précises que celles annotées par des humains.

Module d'apprentissage de reconnaissance globale-local

Ce module permet d'extraire les vecteurs de caractéristiques, qui nous servent à comparer deux détections entre elles, et à produire le vecteur de confiances associées à ces caractéristiques.

Deux types de vecteurs sont crées : ceux associés à une et une seule partie du corps et les vecteurs holistiques qui sont calculées sur l'entierté du corps. Avec les M_i, $i \in \{1, ..., K\}$, nous pouvons créer une carte de probabilité associée au corps entier, M_f, en ne sélectionnant que les probabilités maximales pour chaque pixels, tel que $M_f = max(M_1(h,w), ..., M_K(h,w))$.

Le calcul des vecteurs d'apparence f_i s'effectue via un global weighted average pooling ($GWAP$), défini comme :

$$f_i = \frac{\sum_{h=0}^{H-1}\sum_{w=0}^{W-1} G(h,w)M_i(h,w)}{\sum_{h=0}^{H-1}\sum_{w=0}^{W-1} M_i(h,w)} \tag{2.3}$$

Nous calculons aussi l'apparence globale f_g en effectuant cette fois-ci un simple global average pooling (GAP) sur G. Finalement, f_c est le vecteur de caractéristiques correspondant à la concaténation des f_i. On peut donc différencier les caractéristiques globales : $\{f_g, f_f, f_c\}$ et les caractéirstiques locales : $\{f_1, ..., f_K\}$.

Le cacul de score de visibilité, binaire, est assez simple. Pour les caractéristiques globales, leurs valeur est unitaire. Pour les locales, il suffit qu'un seul pixel dans M_i soit supérieur à une certaine constante (probabilité empirique) λ_v pour que la confiance attribuée à la partie du corps i soit unitaire. Sinon, elle est posée à 0.

Fonction de coût

La fonction de coût totale à optimiser s'écrit :

$$L = \lambda_{pa} L_{pa} + L_{GiLt} \tag{2.4}$$

Où L_{pa} est la fonction définie en 2.1, pondérée par λ_{pa} et L_{GiLt} la fonction du module 3.2.

Sans rentrer dans les détails, nous effectuons une fonction de coût triple [7] modifiée sur les caractéristiques locales, et une fonction d'entropie croisée [12] [18] pour les descripteurs globaux. L_{GiLt} résulte en une combinaison de ces fonctions :

$$L_{GiLt} = L_{id} + L_{tri} \tag{2.5}$$

Résultats

Pour finir avec le BpbReID, différents résultats peuvent être obtenus en sa sortie : des caratéristiques globales, et des locales. Il est important de bien en tenir compte lorsqu'on voudra par la suite comparer deux détections entre elles : voulons-nous donner autant d'importances aux parties locales ou non ?

2.2 Iterative Hypothesis testing

L'algorithme de tracking que nous allons implémenter s'appelle *Iterative Hypothesis Testing* (IHT). Pour le comprendre, il faut se souvenir de l'équation de la formulation du problème 1.1, qui est NP-complet. Les travaux précédents [3], [20], [9] sont parvenus à contourner ce problème en redéfinissant $C(T_i)$, qui ne tient plus en compte que des arêtes qui relient les noeuds consécutifs entre eux :

$$C(T_i) = \sum_{u, v^*(u) \in T_i} w_{u, v^*(u)} \tag{2.6}$$

Où $v^*(u) = \arg\min_{v \in T_i, t_v > t_u}(t_v - t_u)$. Grâce à cette simplification, il suffit ensuite d'appliquer un algorithme qui calcule le set de T_i. Cela peut être fait en utilisant par exemple le *K-shortest path algorithm* (KSP), qui calcule K chemins les plus courts dans un graphe, en minimisant le coût total. Il existe aussi d'autre méthodes qui ne nécessitent pas de connaître K à l'avance.

Le problème avec cette solution et ce genre d'algorithme, c'est que nous n'exigeons plus une cohérence tout le long de la trajectoire T_i. En effet, comme nous ne tenons plus en compte que des arêtes qui relient les noeuds consécutifs, et que la confiance accordée aux caractéristiques d'un noeud (et la disponibilité des caractéristiques) peut varier, le cas illustré sur la figure 2.3 suivant peut survenir : (a) indique le ground truth des trajectoires. La caractéristique des noeuds peut prendre trois valeurs : rouge, bleu, ou gris qui correspond au neutre car le détection est partiellement visible et/ou sa caractéristique est peu sûre. (b) montre le graphe qui est construit dans le cas où est appliqué un algorithme classique de recherche du plus court chemin. (c) montre les plus courts chemins trouvés. Cette méthode est problématique car elle ne tient pas en compte de l'apparence globale de la trajectoire. Le poids entre chaque noeud n'est influencé que par le noeud entrant et sortant. (d) montre la méthode utilisée par IHT. Dans ce cas, les poids entre les noeuds ne sont plus seulement dépendants des noeuds entrant et sortant, mais aussi du noeud de départ. Un noeud clé (de départ) est choisi, le noeud **a** dans notre cas. Le graphe est ainsi modifié en fonction de l'apparence de **a** : les arêtes entrantes dans les noeuds rouges sont pénalisées fortement, les poids des noeuds gris pénalisés faiblement, et aucune pénalité pour les poids des noeuds verts. Grâce à ça, la trajectoire reste cohérente.

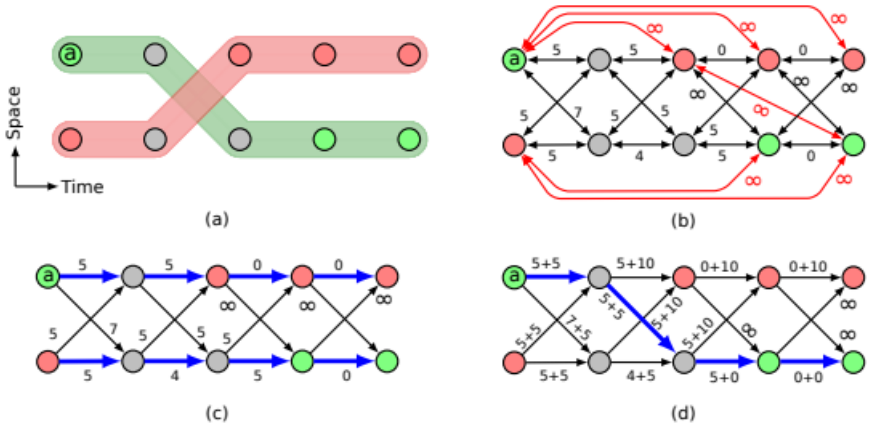


Figure 2.3 – Exemple [1]

En découle le désavantage qu'il n'est plus possible de calculer plusieurs trajectoires en même temps, étant donné que pour chaque noeud de départ, un graphe modifié lui est associé.

Il est important de noter que dans le graphe défini au-dessus, si plusieurs noeuds sont reconnus comme faisant partie de la même trajectoire, alors ils seront agrégés en un seul noeud, dont l'apparence sera la moyenne pondérée de toutes les détections qu'il contient, et son t_{start} et t_{end} correspondront respectivement au t_{start} de la première détection qu'il contient et au t_{end} de la dernière détection qu'il contient. Un noeud représente donc un set de détections, *tracklet*.

Le processus se produit sur plusieurs itérations. A chaque itération, tous les noeuds présents dans le graphe sont investigués : un noeud sélectionné est défini comme *noeud clé*. Son apparence devient alors l'apparence *clé*, et chaque poids des arêtes dans le graphes seront pénalisés ou favorisés en fonction de la similarité des caractéristiques d'apparence des noeuds entrant avec l'apparence clé. Il reste alors à trouver le plus court chemin dans le graphe, qui relie le *noeud clé* et son noeud de fin de trajectoire. Le plus court chemin n'est pas calculé sur le graphe tout entier, mais plutôt sur un graphe tronquée temporellement, qui est un sous-graphe du graphe total, d'une longueur proportionnelle à la taille du *noeud clé*. Sur la figure 2.4 nous observons la fenêtre de longueur δ qui tronque le graphe entier. d_i^j correspond à la détections appartenant à la trajectoire i du ground truth, dans l'image j.

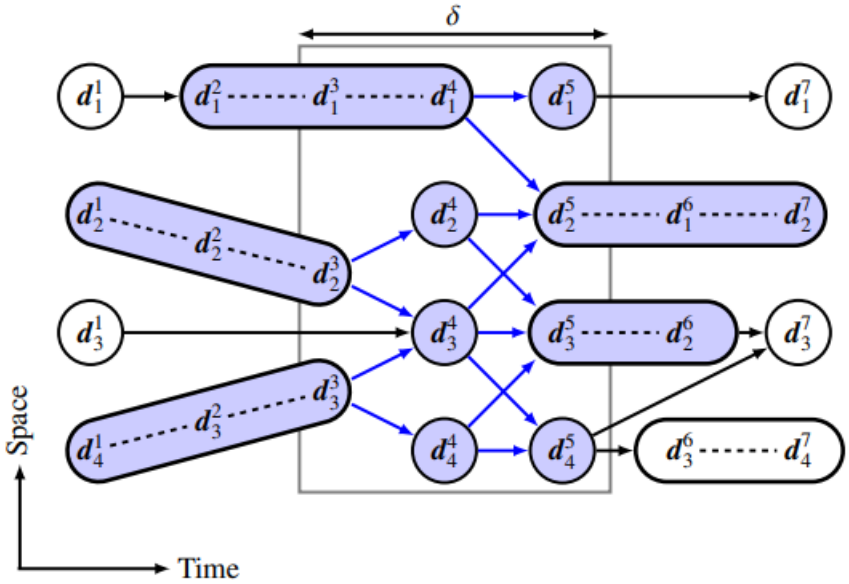


Figure 2.4 – Tracklets [1]

Cette méthode multi-échelle, en outre de permettre un gain en complexité, permet d'investiguer un plus court chemin d'une taille raisonnable par rapport à la taille du *noeud clé*. Plus celui-ci sera grand, plus il est probable que son apparence soit fiable et donc plus il est possible pour l'algorithme de se permettre d'agréger de noeuds dans sa trajectoire.

Une fois ce plus court chemin trouvé,celui-ci est validé ou non, et un nouveau *noeud clé* est sélectionné. A chaque itération, les conditions pour qu'un plus court chemin soit validé ou non, sont très strictes au début mais sont assouplies petit à petit. Cette méthode rend le processus "gourmand", dans le sens où les trajectoires les plus sûres soit extraites en premier, et ce, indépendamment de l'ordre dans lequel les *noeuds clés* sont choisis. Finalement, à la fin de toutes les itérations, on obtient un graphe modifié, dans lequel chaque noeud correspond à une trajectoire finale. S'il reste à la fin n noeuds, notre algorithme aura alors identifié n objets différents dans la vidéo.

2.2.1 Notations

Avant de se plonger dans le pseudo code de l'IHT, il est important de définir quelques notations qui seront utiles à la compréhension de celui-ci :

Détections

t	instant temporel
$\boldsymbol{y}$	position/localisation de la détection
N	nombre de caractéristiques d'apparence
$\boldsymbol{f}_i$	i-ème caractéristique d'apparence, $1 \leq i \leq N$
$\mathcal{F}$	$:= \{\boldsymbol{f}_1, ..., \boldsymbol{f}_N\}$ le set des N caractéristiques d'apparence extraites
$\boldsymbol{c}$	$:= (c_1, ..., c_N)$ vecteur de confiance des caractéristiques
$\boldsymbol{d}$	$:= (t, \boldsymbol{y}, \mathcal{F}, c)$ une détection
$\mathcal{D}^t$	set des détections à l'instant t
$n_{detections}$	nombre de détections dans $\mathcal{V}$

Graphe

$\mathcal{G}$	graphe du set de noeud $\mathcal{V}$, set d'arêtes $\mathcal{E}$, et des poids $\boldsymbol{W}$
$\mathcal{G}^-$	$\mathcal{G}$ inversé temporellement
G_δ	sous-graphe de $\mathcal{G}$, tronqué par $fen\hat{e}tre_\delta$, dont le set de noeud $\mathcal{V}_\delta$, set d'arêtes $\mathcal{E}_\delta$, et des poids $\boldsymbol{W_\delta}$
G_δ^t	graphe à l'instant t
w_{uv}	coût de l'arête $(u, v) \in \mathcal{E}$
w_{uv}^δ	coût total pour aller de la fin du noeud u à la fin du noeud v dans G_δ
$t_v^{(s)}$	temps du départ du noeud v
$t_u^{(e)}$	temps de fin du noeud u
τ_{max}	durée maximale entre laquelle deux noeuds peuvent être reliés
w^*	coefficient de détéction ratée/passée

Hypothesis testing

v_{key}	noeud clé
$\mathcal{R}$	set de noeuds pas encore investigués
κ	constante de proportionnalité de la fenêtre
$\overline{\boldsymbol{f}_i^{(v)}}$	la moyenne de la i-ème caractéristique du noeud v
$D_{key}^{(v)}$	différence globale de similarité d'apparence entre v_{key} et v
$w_v^{(key)}$	coût interne au noeud v dépendant de v_{key}
δ	taille de la fenêtre $fen\hat{e}tre_\delta$
S_b	plus court chemin partant de v_{key} dans G_δ
S_{sb}	second plus court chemin partant de v_{key} dans G_δ
K_1, K_2	paramètres utilisés pour valider S_b

Pour finir, définissons une tracklet comme étant un set de détections, triées dans l'ordre temporel croissant tel que $v = \{\boldsymbol{d}^1, \boldsymbol{d}^2, ..., \boldsymbol{d}^{|v|}\}$ où $|v|$ est la taille de v et

donc $d^{|v|}$ correspond à |v|-ème (et dernière) détection de v.

Comme les arêtes sont dirigées par la temporalité, le graphe est dirigé et acyclique (aucune boucle n'est permise). De manière évidente, et comme déjà expliqué dans le chapitre 1, aucune arrête ne relie les détections présentes dans le même $\mathcal{D}^t$. Cependant, le graphe peut-être inversé de manière globale dans notre algorithme.

2.2.2 Construction du graphe

Dans un premier temps, il faut créer le graphe à partir du set de détections obtenues par le détecteur et enrichies par le ReID. Pour chaque détection à l'instant t, un noeud u lui est associé. Chaque noeud u sera relié par une arête associée à un poids spatio-temporel, uniquement avec les autres noeuds v tel que $0 < t_v^{(s)} - t_u^{(e)} \leq \tau_{max}$. Ce poids w_{uv}^{δ} peut-être calculé à l'aide d'une distance spatio-temporelle. Plusieurs méthodes existent, celle que nous utiliserons est détaillée dans la section 3.3.1. Le graphe ainsi construit est défini $\mathcal{G}_0$. Par la suite, plusieurs noeuds pourront s'associer pour former un super noeud : une tracklet.

2.2.3 Pseudo-code

La fonction 1 est la fonction principale de l'IHT. Le paramètre *dir* permet de changer la direction dans laquelle le processus va effectuer sa recherche de tracklets. L'algorithme comporte 4 sous fonctions que nous allons développer :

- **Schedule**($\mathcal{R}$). Nous trions d'abord $\mathcal{R}$ en fonction du nombre de détections que comporte les noeuds dans $\mathcal{R}$ dans l'ordre décroissant. Le premier élément de la liste est le noeud candidat *noeud clé*. Nous agissons de la sorte car comme dit précédemment, plus une tracklet est longue, plus on peut considérer que ses informations d'apparence sont connues et fiables.
- **HypothesisTesting**($\mathcal{G}, v_{key}, dir, K_1^{(l)}, K_2^{(l)}$). Cette fonction est assez longue et va être développée plus en détail en dessous
- **Simplify**($\mathcal{G}, v_{agg}$) La trajectoire S_b terminant par v_{agg} est fusionnée avec v_{key} pour ne former qu'un seul noeud. Par conséquent, il faut modifier $\mathcal{G}$ car les arêtes reliant les noeuds intérieurs à S_b au reste des noeuds du graphe ne sont plus d'actualité.
- **Relax**($K_1^{(l)}, K_2^{(l)}$). Cette fonction permet d'augmenter progressivement la valeur de K_1 et K_2 et par conséquent rendre plus souple les conditions d'agrégation.

Algorithm 1 Iterative Hypothesis Testing

Require: Graph $\mathcal{G}_0$ and the number of iteration `max_iter`
Ensure: Graph $\mathcal{G}_0$ is transformed into $\mathcal{G}_{\texttt{max\_iter}}$
  procedure IHT($\mathcal{G}_0$, `max_iter`)
    dir $\leftarrow +1$
    Initialize $K_1^{(1)}$ and $K_2^{(1)}$
    for l=1,...,`max_iter` **do**
      Initialize : $\mathcal{R} \leftarrow \mathcal{V}$
      while $\mathcal{R} \neq 0$ **do**
        $v_{key} \leftarrow$ Schedule($\mathcal{R}$)
        $v_{agg} \leftarrow$ HypothesisTesting($\mathcal{G}, v_{key}, dir, K_1^{(l)}, K_2^{(l)}$)
        if $v_{agg} \neq v_{agg}$ **then**
          $\mathcal{G} \leftarrow$ Simplify($\mathcal{G}, v_{agg}$)
        end if
        $\mathcal{R} \leftarrow \backslash v_{agg}$
      end while
      $K_1^{(l+1)}, K_2^{(l+1)} \leftarrow$ Relax($K_1^{(l)}, K_2^{(l)}$)
      dir $\leftarrow$ -dir
    end for
  end procedure

Algorithm 2 Hypothesis testing

Require: $\mathcal{G}, v_{key}, dir, K_1^{(l)}, K_2^{(l)}$
Ensure: v_{agg}
  procedure HypothesisTesting($\mathcal{G}, v_{key}, dir, K_1^{(l)}, K_2^{(l)}$)
    if dir==1 **then**
      $\delta \leftarrow [t_{v_{key}}^{(e)}, t_{v_{key}}^{(e)} + \kappa|v_{key}|]$
    else
      $\delta \leftarrow [t_{v_{key}}^{(s)} - \kappa|v_{key}|, t_{v_{key}}^{(s)}]$
    end if
    $\mathcal{G}_\delta \leftarrow$ GraphHypothesis($\mathcal{G}, \delta, v_{key}$)
    $(S_b, S_{sb}) \leftarrow$ Shortest and second shortest path from v_{key}
    if isUnambiguous(S_b, S_{sb}) **then**
      $\mathcal{G}_\delta^- \leftarrow$ ReverseDirection($\mathcal{G}_\delta$)
      $S_{b'}, S_{sb'} \leftarrow$ Shortest and second shortest path in other direction
      if isUnambiguous($S_{b'}, S_{sb'}$) **then**
        $v_{agg} \leftarrow S_b$
      end if
    else
      $v_{agg} \leftarrow v_{key}$
    end if
    return v_{agg}
  end procedure

Algorithm 3 ambiguity

Require: S_b, S_{sb}
Ensure: is the path valid regards to K_1 and K_2
  procedure isUnambiguous(S_b, S_{sb})
    return $(\text{cost}(S_b) < K_1|\delta|$ and $\frac{\text{cost}(S_b)}{\text{cost}(S_{sb})} < K_2)$
  end procedure

Chapitre 3

Méthodologie

Pour mettre en oeuvre l'algorithme dans des situations réelles et tester ses performances, il faut d'abord l'implémenter. Le tracker a été développé en *Python*, et inclus dans le framework *Pb-track*. La librairie utilisée pour gérer les graphes est *Networkx*.

La subtilité dans l'implémentation de l'IHT réside dans le fait de dimmensionner au mieux les poids de chaque arête du graphe $\mathcal{G}$. Pour ce faire, il faut créer une fonction qui prend en entrée les distances d'apparences et spatio-temporelles, et qui donne en sortie un coût associé à ces distances. [1] nous apporte déjà une fonction bien robuste, mais les hyper paramètres qu'elle comporte varient fortement en fonction du dataset sur lequel nous travaillons.

J'ai également conçu une fonction de poids en m'inspirant de celle existante. L'objectif était de simplifier cette fonction.

Dans ce chapitre, nous allons nous pencher sur l'explication de ces fonctions de poids et leurs hyper-paramètres.

3.1 Ground truth annotation

Une étape cruciale pour la conception des fonction de coûts, leurs débugage et leurs évaluation, fut de créer des ground truths de plusieurs trajectoires dans plusieurs vidéos. Etant limité par le temps, j'ai opté pour un sous ensemble de vidéos venant du dataset `Posetrack21`. J'ai ensuite extraits les détections prédites et leurs caractéristiques car l'annotation du ground truth n'est disponible que pour quelques images de la vidéo. Il est donc important de noter que les performances et métriques de l'algorithme peuvent être influencées par l'imprécision des détections.

Ce sous-ensemble de vidéos comporte 3 situations bien différentes entre-elles, et représentatives de l'entierté du graphe :

1. la vidéo *001735* : Un grand nombre de personnes qui se déplacent de manière fluide et prédictible, sans rotation sur eux-mêmes, dont les vêtements ne se ressemblent pas fort entre eux, comportant quelques longues occlusions. La caméra ne bouge pas
2. la vidéo *002276* : Une vidéo filmant une chaussée comportant des cyclistes, piétons, et voitures. Il y a de longues occlusions, et des vêtements similaires, mais pas de mouvement et pas beaucoup de détections
3. la vidéo *022688* : Une vidéo de rugby, où quelques personnes ont les mêmes habits et couleur de cheveux, la caméra bouge beaucoup et les images sont parfois floues. Il y a beaucoup d'occlusions rapides, ce qui rend cette vidéo à priori la plus difficile à tracker

3.2 Distance

Il existe plusieurs manière de définir une distance entre deux noeuds dans le graphe. Les deux principales classes de distance sur lesquelles nous travaillons sont les distances d'apparence et les distances spatio-temporelles. Pour ce qui est de l'apparence, on utilise le ré-identificateur `BpbReid` pour obtenir des vecteurs d'apparence associés à chaque partie du corps, pour une détection précise.

En sortie du ré-identificateur, chaque détection possède un vecteur de taille $R^{7\times256}$ et un vecteur de score de visibilité de taille $R^{1\times7}$. Comme un noeud comprend en général plusieurs détections, il est logique de créer un nouveau vecteur globale, prenant en compte l'information d'apparence de chaque détections, pour être en mesure de comparer deux noeuds entre eux. La moyenne pondérée a été choisie :

$$\overline{\boldsymbol{f}_i^{(v)}} = \frac{1}{\sum_{t=1}^{|v|} c_{i,t}^{(v)}} \sum_{i=1}^{|v|} c_{i,t}^{(v)} \boldsymbol{f_{i,t}^{(v)}} \tag{3.1}$$

où :

— $\overline{\boldsymbol{f}_i^{(v)}}$ est le vecteur de moyenne pondérée de la i-ème partie de corps associé au noeud v

— $c_{i,t}^{(v)}$ est le score de visibilité associé à la i-ème partie du corps, pour la détection apparaissant au temps t

— $\boldsymbol{f_{i,t}^{(v)}}$ est le vecteur d'apparence de la i-ème partie de corps associé au noeud v, pour la détection apparaissant au temps t

En ce qui concerne les distances spatio-temporelles, il existe plein de méthodes différentes pour effectuer ces mesures. Une méthode assez classique de projection de trajectoire de cible du noeud u vers le noeud v a été choisie :

$$d_{sp-temp} = ||\boldsymbol{y}_v^{(s)} - \boldsymbol{y}_u^{(e)} - \dot{\boldsymbol{y}}_u^{(e)}(t_v^{(s)} - t_u^{(e)})||_2 + w^*(t_v^{(s)} - t_u^{(e)} - 1) \tag{3.2}$$

où :

- l'indice (e) signifie que c'est la dernière détection du noeud. (s) signifie que c'est la première.
- $\boldsymbol{y}$ sont les coordonnées de la détection dans l'image
- $t_v^{(s)} - t_u^{(e)}$ est la durée écoulée entre la fin d'apparition du noeud u et le début d'apparition du noeud v
- w^* est le terme de pénalité associé à une détection manquée

3.3 Fonction de poids

Nous allons à présent nous concentrer sur la fonction de poids qui pondère les arêtes entre chaque noeuds dans le graphe $\mathcal{G}_\delta$. Bien pondérer ces arêtes est une étape déterminante pour le bon fonctionnement de l'algorithme car c'est grâce à cela que nous pourrons au mieux distinguer les noeuds entre eux. Dans un premier temps, nous développerons la fonction de coût originale. Ensuite, nous allons investiguer une variante de cette fonction et voir ses avantages et défauts par rapport à la première.

3.3.1 Fonction de poids originale

Lorsqu'on effectue l'algorithme IHT sur $\mathcal{G}_\delta$, la fonction de coût utilisée pour définir le poids entre deux noeuds du graphe, lorsque la tracklet de départ est *noeud clé* est définie comme :

$$w_{u,v}^\delta = w_v^{key} + w_{u,v} \tag{3.3}$$

où :

- $w_{u,v}^\delta$ est le poids total entre deux noeuds dans $\mathcal{G}_\delta$
- w_v^{key} est le poids de similarité entre le noeud entrant v, et v_{key}. Ce poids appelé "coût interne" est calculé en traversant l'entierté du noeuds v. Il est nécessaire pour éviter que des raccourcis soient formés.
- $w_{u,v}$ est le poids spatio temporel entre la fin du noeud u et le début du noeud v. Il est indépendant de v_{key}

La fonction peut être récrite comme :

$$w_{u,v}^{\delta} = \underbrace{|v|(c_{sp} + D_{key}^{(v)})}_{w_v^{key}} + \underbrace{w^* T + g_{sp}(u,v)}_{w_{u,v}} \tag{3.4}$$

où :

- $|v|$ est la longueur de la tracklet v. Ce facteur permet d'éviter de favoriser le passage par un noeud composé de beaucoup de détections, engendrant un court-circuit, alors qu'un autre chemin complet et correct est envisageable
- c_{sp} est la pénalité due à l'incertitude sur la spatio temporalité (estimée à l'avance). Elle correspond au poids spatio-temporel qui relie les détections consécutives internes d'un noeud
- $D_{key}^{(v)}$ est la distance d'apparence entre la tracklet v et la tracklet key. Elle prend en compte les caractéristiques des deux noeuds qu'elle relie, et la confiance associée à ces caractéristiques
- w^* est le facteur de pénalité associé un une détection manquée.
- T est la différence de temporalité entre deux noeuds, défini comme $T = t_v^{(s)} - t_u^{(e)} - 1$
- g_{sp} est la distance spatio temporelle entre u et v. On peut la voir comme une mesure entre l'emplacement spatial de v et la prédiction de u projetée sur la même image que celle de v

Finalement, pour un maximum de clareté, définissons $D_{key}^{(v)}$, w^* et g_{sp} :

$$D_{key}^{(v)} = \sum_{i=1}^{N} [\alpha_i^{(key)} \alpha_i^v || \overline{\boldsymbol{f}_i^{(key)}} - \overline{\boldsymbol{f}_i^{(v)}} ||_1 + (1 - \alpha_i^{(key)} \alpha_i^{(v)}) w_i^{(fix)}] \tag{3.5}$$

$$w^* = \gamma(c_{sp} + \sum_{i=1}^{N} w_i^{(fix)}) \tag{3.6}$$

$$g_{sp}(u,v) = ||\boldsymbol{y}_v^{(s)} - \boldsymbol{y}_u^{(e)} - \dot{\boldsymbol{y}}_u^{(e)} (t_v^{(s)} - t_u^{(e)})||_2 \tag{3.7}$$

où :

- $\boldsymbol{f}_i$ est le vecteur d'une caractéristique pour un noeud v donnée. Il est de taille $[|v| \times 256]$
- α_i est le score de visibilité/confiance associé à ce vecteur
- $w_i^{(fix)}$ est un paramètre fixé empiriquement
- γ est le paramètre qui pondère la pénalité associée à une détection manquée

Finalement, une légère modification est apportée à la fonction de coût totale. Dans 3.4, il manque un facteur à rajouter devant $g_{sp}(u,v)$ et c_sp. Ce facteur x_{gsp} permet de pondérer l'influence des facteurs spatio-temporels à notre guise. Plus il sera élevé, et plus la spatio-temporalité sera favorisée par rapport à la similarité d'apparence. On peut donc réécrire 3.4 par :

$$w^{\delta}_{u,v} = |v|(x_{gsp}c_{sp} + D^{(v)}_{key}) + \gamma(x_{gsp}c_{sp} + \sum_{i=1}^{N} w_i^{(fix)})T + x_{gsp}g_{sp}(u,v) \tag{3.8}$$

Avant de s'attaquer à la recherche des paramètres optimaux, il faut d'abord rechercher la valeur des paramètres à fixer de manière empirique, $w_i^{(fix)}$ et c_{sp}.

$w_i^{(fix)}$ **:**

Le paramètre $w_i^{(fix)}$ est directement en lien avec la valeur neutre de la distance d'apparence, comme nous pouvons le voir dans l'équation 3.5. En effet, si la confiance totale $\alpha_i^{(key)}\alpha_i^{(v)}$ accordée à la i-ème caractéristique tend vers zéro, le terme $D^{(v)}_{key,i}$ tend vers $w_i^{(fix)}$. Dans ce cas, il faut donner une valeur à $w_i^{(fix)}$ telle que :

$$D_+ = moyenne(||\overline{\boldsymbol{f}_i^{(key)}} - \overline{\boldsymbol{f}_i^{(v)}}||_1),\ \forall (v, key) \in T_j \tag{3.9}$$

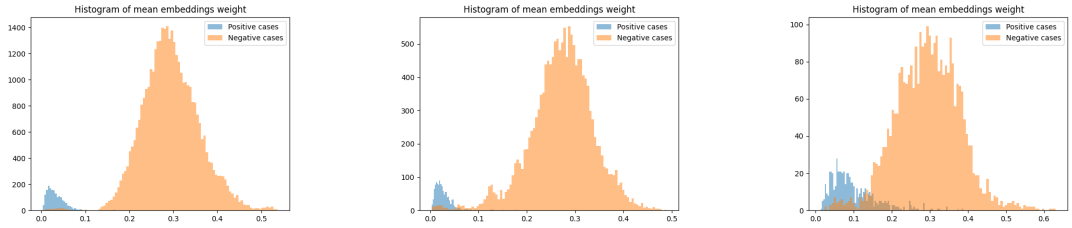
$$D_- = moyenne(||\overline{\boldsymbol{f}_i^{(key)}} - \overline{\boldsymbol{f}_i^{(u)}}||_1),\ \forall u \notin T_j \text{ and } key \in T_j \tag{3.10}$$

$$T_j \in \{T_a\}_{a=1}^{M} \tag{3.11}$$

$$D_+ \leq w_i^{(fix)} \leq D_- \tag{3.12}$$

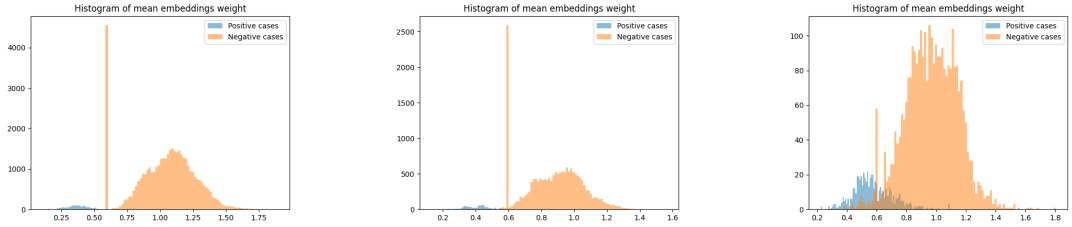
Où $\{T_a\}_{a=1}^{M}$ est le set des trajectoires dont le ground truth a été annoté (voir section 3.1). C'est le set experimental des M trajectoires. D_+ correspond à la moyenne des distances entre une caractéristique d'un *noeud clé* et un noeud v appartenant à la même trajectoire. D_- correspond à la moyenne des distances entre une caractéristique d'un v_{key} et un noeud v n'appartenant pas à la même trajectoire.

A noter que dans ce cas spécifique, nous utilisons une distance de cosinus et non plus un norme d'ordre 1. De plus, je n'ai compté que les caractéristiques ayant une confiance maximale, pour ne pas biaiser les résultats en y ajoutant des vecteurs de caractéristiques peu sûrs. Comme les résultats du tracker dépendent de cette distance, et que les noeuds ne peuvent être aggrégés qu'en faisant tourner l'algorithme, je ne peux mesurer que les distances entre les noeuds du graphe $\mathcal{G}_0$. Cela peut résulter en une légère imprécision pour la suite du développement.



(a) vidéo 1 *(b) vidéo 2* *(c) vidéo 3*

FIGURE 3.1 – *Moyenne des distances d'apparences*



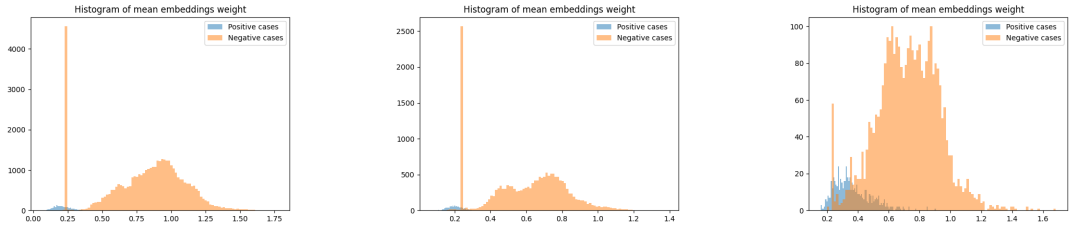
(a) vidéo 1 *(b) vidéo 2* *(c) vidéo 3*

FIGURE 3.2 – *Moyenne sur* $D_{key}^{(v)}$ *quand* $w_i^{(fix)} = 0.1$

Sur la figure 3.1, D_+ est représenté en bleu, et D_- en orange. Nous observons que les deux courbes ont tendance à se chevaucher aux alentours d'une valeur proche de 0.1. Nous en déduisons donc que c'est dans cette région que doit se trouver $w_i^{(fix)}$: en dessous de cette valeur, la plupart des distances entre les caractéristiques correspondent à deux détections faisant partie de la même trajectoire, et inversément une fois le seuil 0.1 atteint.

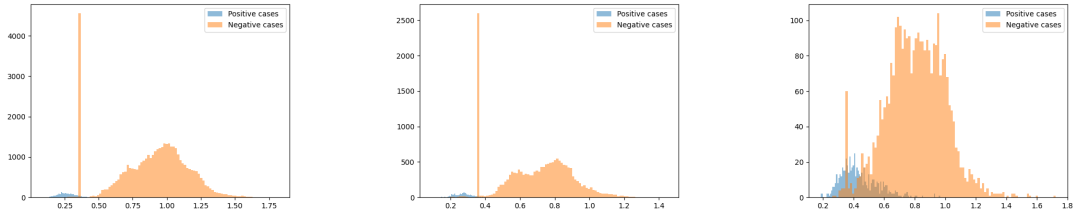
$w_i^{(fix)}$ fixé implique aussi que le terme w^* est fixé. Les valeurs de la première caractéristique (correspondant à l'avant-plan) sont en général bien plus élevées que les 6 autres et peuvent donc pervertir les résultats. J'ai donc fait le choix de ne pas la compter. Nous obtenons donc $w^* = 6 \times 0.1 = 0.6$. En traçant le graphique en batonnets cette fois-ci des valeurs que prennent $D_{key}^{(v)}$ avec les valeurs fixées de $w_i^{(fix)}$, nous voyons sur la figure 3.2 que les deux courbes se séparent à l'emplacement où $D_{key}^{(v)} = 0.6$. Cela correspond bien au neutre comme prédit.

Pour vérifier notre raisonnement, les figures 3.3, 3.4, 3.5 et 3.6 prouvent que le choix d'un $w_i^{(fix)}$ fixé à 0.1 coupe le mieux les deux courbe avec les valeurs neutres. Cependant, nous remarquons qu'avec des valeurs de $w_i^{(fix)}$ plus élevées, certes les neutres commencent à se fondre avec les courbes negatives, mais les courbes se séparent beaucoup mieux entre elles.



(a) vidéo 1 *(b) vidéo 2* *(c) vidéo 3*

FIGURE 3.3 – *Moyenne sur* $D_{key}^{(v)}$ *quand* $w_i^{(fix)} = 0.04$



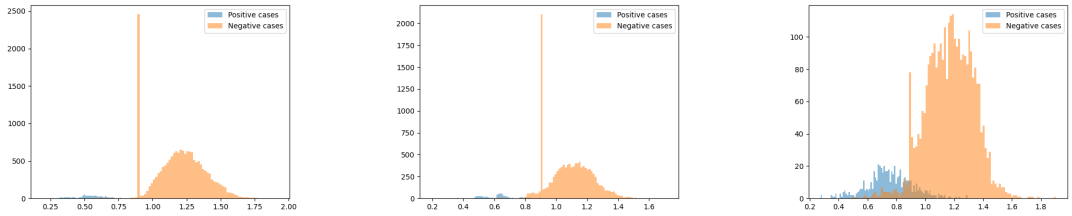
(a) vidéo 1 *(b) vidéo 2* *(c) vidéo 3*

FIGURE 3.4 – *Moyenne sur* $D_{key}^{(v)}$ *quand* $w_i^{(fix)} = 0.08$

Comme nous travaillons avec un procédé gourmand qui commencera par agréger seulement les noeuds qui sont fiables, et donc avec de faibles distances entre eux, nous pouvons espérer que ces agrégations permettent aux noeuds peu fiables de s'agréger plus facilement avec les autres noeuds inlcus dans leurs trajectoires plutôt qu'avec de mauvais noeuds. De plus, alors que pour $w_i^{(fix)} = 0.1$, les courbes de cas positifs et cas négatifs varient fort en fonction des vidéos, il semble qu'elles varient moins avec des valeurs de $w_i^{(fix)}$ plus grandes. Cela va nous être très utile par la suite, car cela permettra d'uniformiser les performances pour tout type de situations sportives. On choisit finalement $w_i^{(fix)} = 0.15$

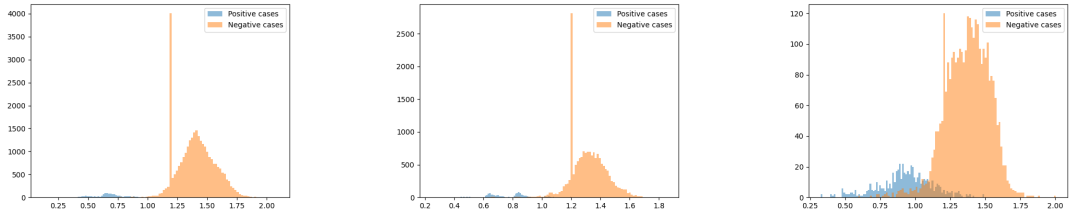
c_sp :

L'estimation de c_{sp} est un peu plus délicate que celle de $w_i^{(fix)}$. Car pour être le plus précis possible, et ne disposant pas d'une estimation directement indiquée par le dataset en question, il faut l'estimer soi-même. Ce paramètre représente le coût moyen spatio-temporel entre deux détections contenues dans un noeud. c_{sp} peut être déterminé en observant la moyenne des g_{sp} entre les noeuds appartenant à la même trajectoire. Dans la section 3.3.2, les graphes 3.8 nous donnent une



(a) vidéo 1 *(b) vidéo 2* *(c) vidéo 3*

FIGURE 3.5 – *Moyenne sur* $D_{key}^{(v)}$ *quand* $w_i^{(fix)} = 0.15$



(a) vidéo 1 *(b) vidéo 2* *(c) vidéo 3*

FIGURE 3.6 – *Moyenne sur* $D_{key}^{(v)}$ *quand* $w_i^{(fix)} = 0.2$

estimation des g_{sp} dans les cas négatifs et positifs. On observe que la moyenne des distances g_{sp} entre noeuds de la même trajectoire se rapproche de 30 dans les deux premières vidéos, la troisième étant beaucoup plus disparate. Il est difficile de décider quelle valeur assigner à c_{sp} dans ce cas, et nous faisons le choix de l'égaler à 30, en espérant que ce choix n'impacte pas trop les résultats par la suite.

Suite :

Lorsque nous allons évaluer nos hyper-paramètres dans le chapitre 4, il faudra faire attention à bien les sélectionner. En effet, ils ne sont pas si indépendant entre eux que ça.

Dans un premier temps, il faut déterminer l'importance qu'aura la spatio-temporalité. Le paramètre x_{gsp} répond à cette tâche. Prenons une situation globale : dans le calcul du plus court chemin dans $\mathcal{G}_\delta$, nous avons 3 chemins possibles partant de v_{key} en t_i vers t_{i+1} :(1) passer par un noeud u qui appartient à la même trajectoire que le *noeud clé*, (2) ne passer par aucun noeud dans l'image au temps t_{i+1}, (3) ou bien passer par un mauvais noeud u. Evidemment, nous voulons prioriser (1) et pénaliser (3). Pour (2), cela dépend : voulons nous à tout prix résoudre des

occlusions, quitte à créer des courts circuits dans le chemin ? Si oui, alors on va le rapprocher de (1). Ou au contraire, nous voulons éviter les courts circuits, et nous allons le rapprocher de (2). Nous pouvons récrire cela sous forme d'inéquation :

$$D_{key}^{(v)}+x_{gsp}(c_{sp}+g_{sp}(key,v)) < \gamma(x_{gsp}c_{sp}+(N-1)w_i^{(fix)}) < D_{key}^{(u)}+x_{gsp}(c_{sp}+g_{sp}(key,u)) \tag{3.13}$$

Où $D_{key}^{(v)}$, $D_{key}^{(u)}$, $g_{sp}(key,v)$ et $g_{sp}(key,u)$ sont des variables aléatoires dont les distributions ont été anaylsées plus haut. Via ces estimations, nous pouvons prédire très approximativement que le coût de (1) a peu de probabilité de dépasser une certaine valeur, et le coût de (2) a peu de probabilité d'être inférieur à une autre. Grâce à cela, une estimation de plages de valeur possible sur γ peut être faite. Nous remarquons donc que nous ne pouvons pas donner n'importe quelle valeur à γ sous peine de ne plus respecter 3.13. Finalement, les paramètres K_1 et K_2 sont les derniers paramètres à ajuster. K_1 dépend directement du coût de (1) et par conséquent du choix de x_{gsp}. K_2, lui, dépend de x_{gsp} et de γ.

3.3.2 Variante de la fonction originale

Cette fonction a été implémentée chronologiquement avant la fonction originale, dans le but de démarrer les analyses avec une vesion simplifiée des poids. Dans la fonction originale, deux points compliquent la tâche :

- les terme détections manquées w^* et de similarité d'apparence $D_{key}^{(v)}$ ne sont pas indépendants entre eux, à cause du terme $w_i^{(fix)}$. Il faut aussi déterminer la relation de proportionnalité entre g_{sp} et $D_{key}^{(v)}$, via x_{csp}. Ce coefficient impacte directement g_{sp} qui est aussi présent dans w^*
- Les valeurs à assigner à K_1 et K_2 sont compliquées à déterminer, que ce soit pour des valeurs fixes ou variables

Cette nouvelle méthode tente de simplifier ces problèmes en redéfinissant les termes de détections manquées, de similarité et de spatio-temporalité.

Conception

La nouvelle fonction de coût s'écrit comme :

$$w_{u,v}^{\delta} = w^*T + |v|f(D_{key,th}^{(v)} + g_{sp,th}) \tag{3.14}$$

Où les termes diffèrent de 3.4 :

- w^* est le poids associé à une détection manquée

— $f()$ est une fonction qui permet de limiter la valeur de $D^{(v)}_{key,th} + g_{sp,th}$ entre 0 et 1

— $D^{(v)}_{key,th}$ est la distance de similarité d'apparence entre deux noeuds. Elles est calculée via une fonction venant du framework de `BpbReid`. Elle calcule la distance pondérée entre les f_f et f_i de chaque noeuds

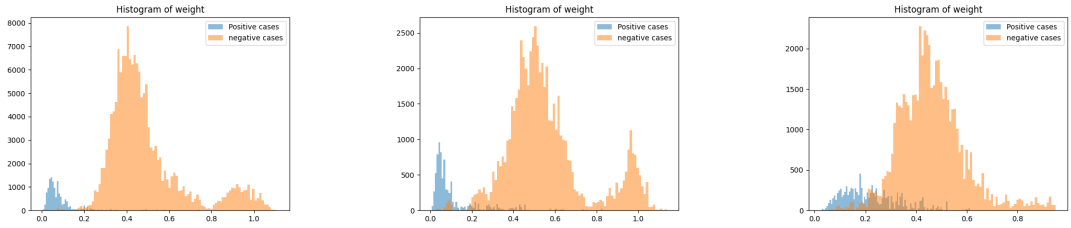
$$D^{(v)}_{key} = \frac{\sum_{i \in \{f,1,\dots,K\}} (c_i^{key} c_i^v cos\_dist(f_i^{key} f_i^v))}{\sum_{i \in \{f,1,\dots,K\}} (c_i^{key} c_i^v)} \tag{3.15}$$

avec c_i^j les scores de visibilité de chaque caractéristique i appartenant au noeud j. Ces distances sont ensuite transformées pour être égalisées à des valeurs appartenant au set discret $\{0, 0.5, 1\}$, via une fonction en escalier, où les valeurs de $D^{(v)}_{key}$ inférieures à D_{neutre} donnent un $D^{(v)}_{key,th} = 0$, les valeurs de $D^{(v)}_{key}$ entre D_{neutre} et $D_{neutre} + 0.1$ sont assignées à 0.5 et le reste à 1.

— $g_{sp,th}$ est le même terme spatio-temporel tel que défini en 3.7, mais transformé pour être compris entre un $-\frac{\alpha_{gsp}}{x}$ et $\frac{\alpha_{gsp}}{x}$, où α_{gsp} et x seront définis plus loin

Passons à présent en revue les différents termes et leurs signification. Le paramètre w^* a la même signication que dans la fonction de coût totale, mais pas la même définition. En fait, il est trouvé empiriquement.
Le fait que $D^{(v)}_{key,th}$ se limite à trois valeurs n'est pas sans raison : en regardant les graphes des distributions des $D^{(v)}_{key}$ [1] sur la figure 3.7 on peut noter qu'il se distingue trois zones : en dessous de $D^{(v)}_{key} = 0.2$, la plupart des distances correspondent à celles entre deux noeuds appartenant à la même trajectoire, et très peu de distances entre deux noeuds qui ne font pas partie de la même trajectoire. Entre 0.2 et 0.3, on note une zone tampon. Les cas positifs et négatifs se mélangent. Au dela de 0.3, la plupart des distances correspondent à des cas négatifs. Aux zones positive, neutres et négatives, sont associées respectivement les valeurs 0, 0.5 et 1. Cette méthode permet de favoriser les paires de noeuds dont nous sommes sûrs qu'ils se ressemblent, et de pénaliser les paires de noeuds fort différents. Les noeuds ambigüs ne sont ni pénalisés ni favorisés.

1. Les distributions de $D^{(v)}_{key}$ différent de celles calculées précedemment. Cela est dû au fait que j'ai récupéré directement la distance calculée pondérée par *BpbReid*. Dans la fonction originale, c'est un peu différent car j'avais aussi besoin des facteurs de confiances associés à chaque caractéristiques, et je ne pouvais donc pas directement réutiliser cette distance



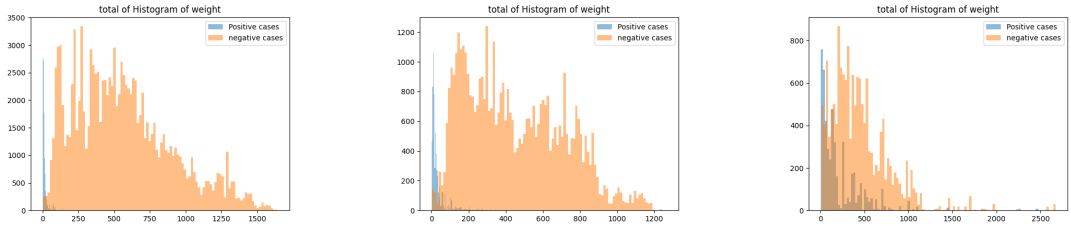
(a) vidéo 1 *(b) vidéo 2* *(c) vidéo 3*

FIGURE 3.7 – Moyenne sur $D^{(v)}_{key}$

Sur les graphes 3.8, sont affichées les distributions des g_sp. L'ordre de grandeur de cette distance est supérieur à celui des apparences, il faut donc remédier à cela. De plus il est possible qu'une paire de noeuds n'ayant pas le même gtID ait une distance $D^{(v)}_{key,th}$ nulle, car les deux cibles se ressemblent. Nous voulons donc les différencier en leur rajoutant un coût associé à g_{sp} et ainsi augmenter le poids qui les relie. Dans une autre situation, il est possible que deux noeuds ayant le même gtID, soient considérés comme neutre à cause du manque de visibilité. Dans ce cas là, nous voulons utiliser la spatio-temporalité pour diminuer le coût total grâce à g_{sp}. Pour ce faire, $g_{sp,th}(u, v)$ est définie comme :

$$g_{sp,th}(u,v) = \frac{max(g_{sp}(u,v),) - \alpha_{gsp}}{x \cdot \alpha_{gsp}} \tag{3.16}$$

où α_{gsp} est le seuil estimé pour lesquelles les distances g_{sp} au-delà de ce seuil sont déterminées comme lointaines, et donc peut susceptibles de correspondre à une distance entre deux noeuds de la même trajectoire. Il se situe idéalement là ou se séparent au mieux les cas positifs des cas négatifs. x est le paramètre qui va pondérer l'importance qu'aura les distances spatio-temporelles par rapport aux apparences.



(a) vidéo 1 *(b) vidéo 2* *(c) vidéo 3*

FIGURE 3.8 – Moyenne sur g_{sp}

Finalement, nous redéfinissons une limite entre 0 et 1 pour la valeur finale associée à l'apparence et au coût spatio-temporel, pour éviter d'avoir des poids négatifs. La fonctions $f()$ sert à cela dans l'équation 3.14. Même si borner le coût maximal à 1 n'est en vérité pas nécessaire.

Validation de la nouvelle fonction de coût

Nous allons observer différents cas de figure possibles de configurations de trajectoires dans le graphe et observer comment la fonction de coût y réagit théoriquement. Pour cela définissons les possibles configurations de chemin qui peuvent être empruntés dans $\mathcal{G}_\delta$.
Appelons X_1 et X_2 la distribution des poids totaux (sans compter le terme w^*T) dans les cas positifs et négatifs. Ces deux courbes sont idéalement le plus éloignées l'une de l'autre mais se chevauchent dans une zone X_{th}, proche du neutre, en 0.5. C'est cette valeur cruciale qui permettra d'estimer si un noeud doit être compris dans le chemin reliant v et *noeud clé*, si le poids total d'apparence et spatio-temporel est inférieur ou supérieur à celui-ci.
Dans le chemin correct, nous ne passons que par des noeuds dont le poids de l'arête entrante est inférieure au poids du neutre. Inversément, dans le cas où le poids est supérieur au poids du neutre, nous décidons de faire un saut temporel. Ce poids est défini par la valeur X_{th} et par conséquent, $w^* = X_{th} = 0.5$ [2].

Chemin correct vs chemin incorrect Tout d'abord, citons les différents chemins qui peuvent exister dans le graphe $\mathcal{G}_\delta$:

- Le chemin "positif" : Il ne passe que par des noeuds ayant un poids X_1. Il peut aussi ne pas nécessairement être présent sur chaque image, si il y a des occlusions
- Le chemin "négatif" : C'est un chemin alternatif au positif, il contient au moins un noeud avec un poids X_2 ou bien au moins une fausse occlusion (c'est à dire qu'il estime qu'il y a une occlusion alors qu'il y a en réalité une détection positive à cet instant)

Appelons P_p le Positive Path et P_n le Negative Path. Définissons aussi de nouvelles variables associé à différents coefficients (compris entre 0 et 1) :

- α : le taux d'occlusions dans P_p
- A : le set d'indices (numéro d'image dans $graph_\delta$) pour lequel le chemin P_p a une occlusion

2. En vérité, il sera très rare que la distance totale entre deux noeuds de la même trajectoire soit égale pile à 0.5. En général, $g_{sp,th}$ permet de réduire ce coût. Il serait donc plus judicieux de tester les différentes valeur de w^* et voir lesquelles donnent de meilleurs résultats. Malheureusement, par manque de temps, je n'ai pas su le faire

— $\overline{A}$: le set d'indices Pour lequel le chemin P_p passe par X_1

— β : le taux de X_2 dans P_n

— B : le set d'indices pour lequel le chemin P_n passe par X_2

— ν : le taux de X_1 dans P_n

— Nu : le set d'indices pour lequel le chemin P_n passe par X_1

— ω : le taux de fausses occlusions dans P_n

— O : le set d'indices pour lequel le chemin P_n a une fausse occlusion

— θ : le taux d'occlusions correctes dans P_n

— T : le set d'indices pour lequel le chemin P_n a une vraie occlusion

Nous pouvons ainsi donner la formule de P_p et P_n :

$$P_p = \sum_{a \in \overline{A}} X_{1_a} + \sum_{\in A} w^* \tag{3.17}$$

$$P_n = \sum_{b \in B} X_{2_b} + \sum_{n \in Nu} X_{1_n} + \sum_{o \in O} w^* + \sum_{t \in T} w^* \tag{3.18}$$

Nous posons l'hypothèse que tous les termes X_{1_a} ont la même distribution de probabilité. Par conséquent, simplifions la somme associée à ce terme :

$$\sum_{a \in \overline{A}} X_{1_a} = \sum_{i=1}^{|\delta|(1-\alpha)} X_1 \tag{3.19}$$

Avec :

$$\frac{\sum_{i=1}^{|\delta|(1-\alpha)} X_1}{|\delta|(1-\alpha)} = \mu^{\delta}_{X_1,P_p} \tag{3.20}$$

Où $\mu^{\delta}_{X_1,P_p}$ est la moyenne des valeurs de X_1 sur le chemin emprunté. Réécrivons l'équation 3.19 comme :

$$\sum_{a \in \overline{A}} X_{1_a} = |\delta|(1-\alpha)\mu^{\delta}_{X_1,P_p} \tag{3.21}$$

Grâce à ce raisonnement, nous pouvons écrire l'inéquation entre le chemin le plus court et le second chemin le plus court :

$$S_b < S_{sb}$$

$$\Leftrightarrow$$

$$|\delta|(1-\alpha)\mu^{\delta}_{X_1,P_p} + |\delta|\alpha w^* < |\delta|\beta\mu^{\delta}_{X_2,P_n} + |\delta|\nu\mu^{\delta}_{X_1,P_n} + |\delta|(\theta+\omega)w^*$$

Voyons maintenant dans quelles conditions l'un ou l'autre noeud sera agrégé. Pour ce faire, il faut que les conditions sur K_1 et K_2 soient respectées.
K_1 :

$$|\delta|(1-\alpha)\mu^{\delta}_{X_1,P_p} + |\delta|\alpha w^* < |\delta|K_1$$
$$\Leftrightarrow$$
$$(1-\alpha)\mu^{\delta}_{X_1,P_p} + \alpha w^* < K_1$$

La contrainte sur K_1 permet de faire en sorte qu'il n'y ait pas n'importe quel chemin qui soit considéré comme valide, et se focalise sur le poids total du chemin. Nous pouvons voir $K_1|\delta|$ comme une limite à la valeur moyenne que doit valoir le chemin le plus court. Le fait que K_1 soit incrémenté petit à petit, tout en commençant par une valeur très faible, assure qu'uniquement les chemins les plus sûrs seront agrégés en au début et éviter ainsi que de mauvaises trajectoires soient établies dès le début, ce qui engendrerait de grosses baisses de performances.

Remarquons donc que pour un α et un K_1 donné, la condition sera respectée ou non en fonction de la valeur moyenne des poids entre chaque noeuds positifs. Forcément, plus K_1 sera faible, plus $\mu^{\delta}_{X_1,P_p}$ devra également l'être. Par exemple, si $\alpha = 0.2$ (donc 20% d'occlusions) et $K_1 = 0.4$, $\mu^{\delta}_{X_1,P_p}$ devra être inférieur à 0.375 pour que la condition soit validée.

Investiguons maintenant K_2 :

$$\frac{|\delta|(1-\alpha)\mu^{\delta}_{X_1,P_p} + |\delta|\alpha w^*}{|\delta|\beta\mu^{\delta}_{X_2,P_n} + |\delta|\nu\mu^{\delta}_{X_1,P_n} + |\delta|(\theta+\omega)w^*} < K_2$$
$$\Leftrightarrow$$
$$\frac{(1-\alpha)\mu^{\delta}_{X_1,P_p} + \alpha w^*}{\beta\mu^{\delta}_{X_2,P_n} + \nu\mu^{\delta}_{X_1,P_n} + (\theta+\omega)w^*} < K_2$$

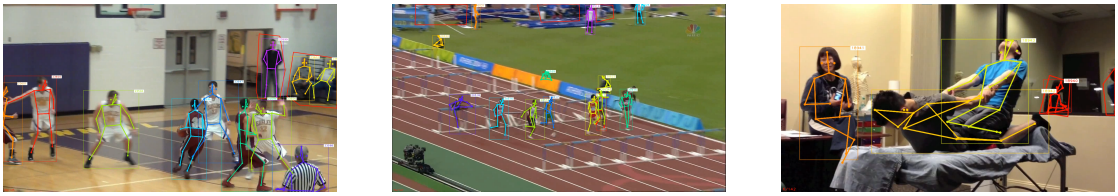
Cette fois-ci la condition sur K_2 impose que le ratio entre le chemin le plus court et le deuxième chemin le plus court soit inférieur à K_2. De la même manière que pour la condition d'au dessus, nous pouvons estimer si la condition sera satisfaite ou non , en connaissant à l'avance les occlusions, mauvais chemins pris etc. Ce raisonnement, bien que peu utile pour déterminer précisemment les hyper-paramètres du tracker, nous aide néanmoins à pouvoir estimer à la louche quelless valeurs ceux-ci doivent prendre selon des cas de figure précis.

Chapitre 4

Evalutations expérimentales

4.1 Base de données

La base de données sur lequel nous travaillons se nomme `Posetrack21` [6] et permet l'évaluation de suivi d'objets ainsi que de l'estimation de poses des objets. Plus précisemment, les images capturées proviennent toutes de situations sportives. Il contient 593 vidéos pour l'entrainement et 170 vidéos pour l'annotation. Contrairement au `MOTchallenge` [5], le dataset le plus utilisé dans la littérature, où la diversité de pose est assez faible, `Posetrack21` offre une large variété de pose du corps humain dans l'espace dans étant donné qu'il couvre un large domaine de sports. En outre, les poses sont annotées dans le ground truth et permettent donc un bon entrainement du modèle `BpbReid`. Il est également compatible avec l'évaluateur du benchmarck de `MOTChallenge`. La figure 4.1 montre la diversité des vidéos sportives que `Posetrack21` contient



(a) Basketball *(b) 110m haies* *(c) Massage sportif*

FIGURE 4.1 – Images extraites de `Posetrack21`

4.2 Détermination des hyper-paramètres

Dans cette section, nous allons nous pencher sur les effets qu'ont les hyper-paramètres de la fonction de poids sur les métriques sélectionnées. Pour ces éva-

luations, le dataset va être séparé en deux parties : le set de validation et set de test (le set d'entrainement étant déjà utilisé pour entrainer les poids du modèle `BpbReid`). Nous allons donc faire tourner l'algorithme sur le set de validation, comparer les métriques et observer les situations critiques, pour ensuite sélectionner les meilleures paramètres. Les métriques finales seront alors évaluées sur le set de test, pour ne pas sur-ajuster nos résultats sur le set de validation.

Par manque de temps, le set de validation ne comporte que trois vidéos : celle sur lesquelles nous avons déjà travaillé en section 3. Ces trois vidéos sont fort différentes et ensembles, représentent bien le reste du dataset.

Pour tester les hyper-paramètres, la méthode optimale est de procéder via une grille de recherche. Il faut spécifier une liste de toutes les valeurs possibles pour chaque paramètre. L'algorithme de grille de recherche se charge d'évaluer les métriques du modèle pour chaque combinaison d'hyper paramètres possibles. La combinaison qui maximise les performances du modèle peut être ainsi trouvée.

Le problème avec ce genre de méthode est qu'elles sont très gourmandes en terme de complexité. Evaluer C_n^2 (n le nombre d'hyper paramètres) combinaisons sur le modèle serait beaucoup trop long. J'ai donc décidé de faire l'hypothèse que les hyper paramètres sont indépendants entre eux, ce qui n'est pas totalement vrai et qui m'empêchera d'atteindre de réellement bons résultats.

Pour chaque vidéo, je fais varier un certain paramètre tout en fixant les autres. J'affiche ensuite les résultats obtenus en fonction de plusieurs métriques obtenues, dans des tableaux.

Une information importante à prendre en compte, est qu'un certain nombre de détections sont mal créées par le détecteur. Il arrive en effet que le détecteur comprenne qu'une personne se situe dans une certaine région, mais la délimite très approximativement. Dans ce cas là, la superposition de cette détection prDet avec le gtPred est inférieure à un certain seuil. Par conséquent, même si l'algorithme de suivi parvient à agréger correctement en une trajectoire les prDet, la détection mal délimitée engendrera un FN, car aucun prDet ne sera associé au gtDet correspondant, et un FP, car aucun gtDet ne sera associé au prDet correspondant.

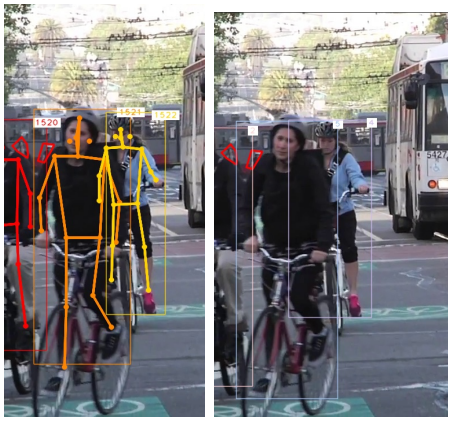


FIGURE 4.2 – La cycliste en arrière plan n'est pas délimitée par le même rectangle du ground truth de la détection prédite. Les détections ne se superposent pas assez ce qui cause un FN et FP

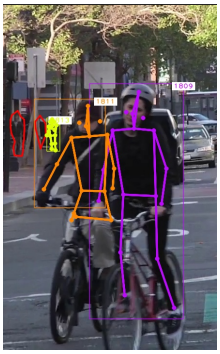


FIGURE 4.3 – Erreur de localisation de la détection 1811

4.2.1 IHT avec fonction de poids originale

Le point de départ où les mesures ont été prises, est ($\gamma = 1.4$, $x_{gsp} = 166$, $\kappa = 3$, $\tau_{max} = 10$). Quand chaque hyper paramètre est testé, les autres sont posés à ces valeurs.

γ :

γ	1				1.3				1.6				1.9			
Vidéo	1	2	3	Moyenne	1	2	3	Moyenne	1	2	3	Moyenne	1	2	3	Moyenne
IDF1	66.5	57.5	25.1	59.4	73.0	53.7	57.3	64.3	70.0	46.6	49.7	59.3	63.0	52.9	55.2	58.5
FP	93	217	2	312	189	351	13	553	191	348	18	557	187	343	16	546
FN	435	305	236	976	408	273	71	752	408	274	58	740	408	274	57	739
IDSW	24	12	17	53	11	19	14	44	11	13	19	43	17	14	15	46
AssA	51.78				53.88				47.76				45.4			
Frag	51.78				52.89				46.85				44.23			

Figure 4.4 – Variations de γ

Un petit γ aura tendance à provoquer des courts-circuits dans le graphe, car la pénalité de saut de détection sera trop faible. Un grand γ empêche de résoudre les occlusions, car la pénalité de saut est trop élevée. Dans les deux cas, cela provoque des IDSW. Ces résultats sont visibles sur le tableau 4.4. Les résultats surprenamment bons dans la vidéo 2 pour un $\gamma = 1$ sont à discuter : il est courant de ne pas attribuer d'ID à des tracklets très courtes (inférieures à 3 en général). Comme beaucoup de courts-circuits sont créés dans cette situation, peu d'ID sont attribuées. Malheureusement, comme expliqué dans la section précédente, des détections trop imprécises peuvent être détectées et associées correctement, mais cela engendrera un FP et FN. Cette vidéo possède particulièrement beaucoup de détections mal localisées. Dans le reste de l'analyse, nous prendrons des pincettes quant aux résultats de cette vidéo.

FIGURE 4.5 – Un faible γ provoque des court-circuits, et par conséquent des tracklets trop courtes pour avoir un ID apparaissent

x_{csp} :

x_{csp}	100				150				200				250			
Vidéo	1	2	3	Moyenne	1	2	3	Moyenne	1	2	3	Moyenne	1	2	3	Moyenne
IDF1	71.9	48.7	48.6	61.0	73.5	51.7	55.0	63.6	69.8	46.6	64.8	61.1	67.4	52.0	70.8	62.5
FP	188	342	12	542	190	347	16	553	193	356	17	566	190	358	20	568
FN	408	276	99	783	407	272	62	741	408	273	57	738	407	271	57	735
IDSW	11	13	20	44	10	12	21	42	12	22	14	48	14	20	8	42
AssA	50.55				52.66				51.06				52.76			
Frag	49.65				51.33				50.51				51.34			

FIGURE 4.6 – Variations de x_{csp}

Pour de faibles valeurs de x_{csp}, le poids spatio-temporel est beaucoup plus important dans l'équation de coût totale. Si ce poids est trop élevé, nous perdons tout l'intérêt de travailler avec des apparences. De plus, les situations avec beaucoup de mouvements brusques et donc des répartitions de g_{sp} plus disparates, auront des résultats encore plus médiocres. Augmenter fortement x_{csp} est fort bénéfique à ces situations là. Mais pour les autres vidéos, l'information spatio-temporelle reste trop importante que pour la diminuer fortement.

τ_{max}

τ_{max}	2				6				10				14			
Vidéo	1	2	3	Moyenne	1	2	3	Moyenne	1	2	3	Moyenne	1	2	3	Moyenne
IDF1	71.8	40.7	58.8	59.4	68.5	51.0	66.5	62.2	69.8	46.6	64.8	61.1	70.1	48.8	65.2	62.1
FP	188	345	16	549	192	356	15	563	193	356	17	566	191	358	16	565
FN	410	272	57	739	407	271	57	735	408	273	57	738	407	273	57	737
IDSW	12	22	11	45	11	20	10	41	12	22	12	46	12	20	14	46
AssA	49.65				53.99				51.54				51.77			
Frag	49.45				53.47				51.51				50.67			

FIGURE 4.7 – Variations de τ_{max}

Si une occlusion est plus grande que τ_{max}, alors celle-ci ne pourra pas être résolue et une fragmentation sera engendrée. A contrario, si τ_{max} est plus grand que cette occlusion, il y a possibilité de résoudre l'occlusion. Ces effets ne varient pas en fonction du type de vidéo.

4.2.2 κ

κ	1				2				4				8			
Vidéo	1	2	3	Moyenne	1	2	3	Moyenne	1	2	3	Moyenne	1	2	3	Moyenne
IDF1	69.2	58.4	59.5	64.3	64.9	56.0	63.2	61.6	74.2	53.4	50.8	64.0	63.8	33.1	50.4	51.5
FP	188	347	14	549	191	350	17	558	191	354	16	561	193	352	9	554
FN	408	272	63	743	408	276	63	747	406	271	63	740	410	278	114	802
IDSW	10	9	18	37	14	12	14	40	10	16	19	45	25	32	12	69
AssA	53.28				48.98				53.25				41.64			
Frag	51.99				47.76				52.89				41.11			

FIGURE 4.8 – Variations de κ

Si κ est trop grand, le graphe $\mathcal{G}_\delta$ sera lui aussi trop grand : les tracklets qui s'associent aux premières itérations permettent de constituer des noeuds fiables par la suite. Le problème ici, c'est qu'à cause des occlusions et manque de fiabilité

des apparence, les tracklets tardent à s'associer ensemblent, et en résulte un grand nombre d'erreurs.

4.2.3 set de résultats

4.2.4 IHT avec fonction de poids revisitée

Le point de départ où les mesures ont été prises, est ($D_{neutre} = 0.24$, $\alpha_{gsp} = 150$, $x = 2$, $\tau_{max} = 20$ et $\kappa = 3$). Quand chaque hyper paramètres est testé, les autres sont posés à ces valeurs.

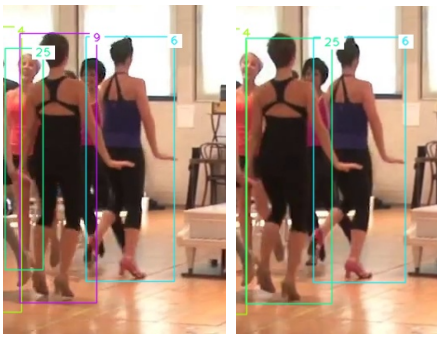
D_{neutre} **:**

La table 4.14 montre les performances du tracker lorsque D_{neutre} varie. Les distributions des $D_{key}^{(v)}$ varient fortement en fonction du type de scène, comme nous l'avons vu sur les graphes 3.7 ce qui aura pour effet de favoriser l'un ou l'autre type de situation. Augmenter D_{neutre} aura pour effet de dimuner les performances globales (IDF1 et IDSW) des vidéos où les caractéristiques de chaque détections sont fort visibles. Au contraire, l'effet inverse se produit pour les vidéos à caractéristiques peu sûres. Nous remarquons aussi que l'augmentation du seuil D_{neutre} a pour effet d'augmenter les FP et dimuner les FN. Plus le seuil augmente, plus le nombre de distances entre deux noeuds de la même trajectoire auront de chance de passer de 0.5 à 0, ou de 1 à 0.5. Par conséquent, il y aura plus de probabilités que ces noeuds soient agrégés ensemble et donc moins de FN. Le contre effet est que les distances entre deux noeuds n'appartenant pas à la même trajectoire voient aussi leurs $D_{key,th}^{(v)}$ diminuer, ce qui résulte en plus de FP.

D_{neutre}	0.15				0.22				0.29				0.36			
Vidéo	1	2	3	Moyenne	1	2	3	Moyenne	1	2	3	Moyenne	1	2	3	Moyenne
IDF1	71.8	48.1	39.7	59.7	68.0	49.0	51.6	59.4	69.7	50.3	57.8	61.4	49.7	49.0	61.2	50.9
FP	178	334	10	522	182	336	12	530	183	338	12	533	189	346	13	548
FN	412	281	81	774	407	279	65	751	426	277	58	761	411	273	60	744
IDSW	19	22	23	64	22	16	19	57	23	23	18	64	80	19	18	117
AssA	51.38				51.48				53.73				41.24			
Frag	51.32				51.45				53.49				39.91			

Figure 4.9 – Variations de D_{neutre}

Les figures 4.10 et 4.11 illustrent nos propos. Dans le cas de la première vidéo, un petit D_{neutre} empêche une mauvaise association de la dame en noire. A l'inverse, dans les figures 4.13, 4.12 nous remarquons qu'une personne partiellement cachée par une autre, peut quand même être correctement associée avec un D_{neutre} plus grand. Ce n'est plus le cas pour un petit D_{neutre}.



(a) image 16 *(b) image 17*

FIGURE 4.10 – Image extraite de la 1ère vidéo : le seuil $D_{neutre} = 0.36$ est trop haut, des détections qui ne se ressemblent pas s'associent et cause des IDSW

(a) image 16 *(b) image 17*

FIGURE 4.11 – Image extraite de la 3e vidéo : le seuil $D_{neutre} = 0.15$, les 2 échouent à s'associer correctement. La trajectoire se fragmente.

(a) image 89 *(b) image 91*

FIGURE 4.12 – Image extraite de la 3e vidéo : le seuil $D_{neutre} = 0.36$ permet une association correcte.

(a) image 89 *(b) image 91*

FIGURE 4.13 – Image extraite de la 1ère vidéo : le seuil $D_{neutre} = 0.15$, les 2 détections s'associent correctement

α_{gsp} :

α_{gsp}	50				100				150				200			
Vidéo	1	2	3	Moyenne	1	2	3	Moyenne	1	2	3	Moyenne	1	2	3	Moyenne
IDF1	75.7	52.9	54.5	65.2	70.6	53.2	54.8	62.6	72.6	51.6	60.3	63.8	64.1	46.0	63.7	57.8
FP	182	344	12	538	182	346	14	542	186	347	16	549	184	341	16	541
FN	408	275	85	768	417	274	61	752	409	274	57	740	410	272	57	739
IDSW	9	16	18	43	23	19	20	62	25	20	13	58	28	26	10	64
AssA	54.05				51.91				52.72				48.26			
Frag	53.11				50.9				50.52				46.59			

FIGURE 4.14 – Variations de α_{gsp}

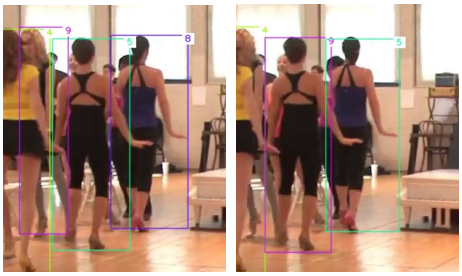
L'effet qu'a α_{gsp} sur les performances est très similaire à D_{neutre} et ce n'est pas anodin. Il permet de déterminer jusqu'à quelle valeur de g_{sp} la distance spatio-temporelle entre les deux noeuds peut être considérée comme acceptable si les deux noeuds font partie de la même trajectoire. Il a aussi un effet de seuil. Comme les vidéos 1 et 2 possèdent des détections qui se déplacent de manières assez prévisibles, les courbes des g_{sp} des cas positifs et négatifs ne se croisent pas vraiment. Par contre, la 3e vidéo est beaucoup plus imprévisible, ce qui entrelace les deux courbes.

Même si elles sont distinguables, les courbes de g_{sp} sont assez proches les une des autres dans les deux première vidéo. Par conséquent, augmenter α_{gsp} aura pour effet de dimuner encore plus la distance originale entre les valeurs limites des deux courbes.

Prenons un exemple : si deux détections sont proches et ne font pas partie de la même trajectoire, mais que leurs distance spatio-temporelle $g_{sp} = 80$, et que la distance moyenne entre deux noeuds de même trajectoire est $c_{sp} = 30$, et que $\alpha_{gsp} = 50$, alors on peut déterminer la distance spatio temporelle transformée :

$$g_{sp,th} = \frac{80 - 50}{2 \times 50} = 0.3 \tag{4.1}$$

Si les deux noeuds ont une distance d'apparence neutre entre eux, le coût total vaudra $w_{u,v}^{(key)} = 0.3 + 0.5 = 0.8 > w^*$. Les deux détections ont peu de chance de s'associer. Mais si $\alpha_{gsp} = 200$, la distance transformée vaudra alors -0.3. La distance totale sera alors très faible et il sera probable que les deux détections s'associent. La fig 4.15 illustre nos propos. Les deux dames ont une similarité neutre, et des mouvement proches, provoquant un IDSW.



(a) image 11 *(b) image 12*

FIGURE 4.15 – Image extraite de la 1ère vidéo : le seuil $\alpha_{gsp} = 200$ est trop élevé

x :

x	0.5				1				1.5				2			
Vidéo	1	2	3	Moyenne	1	2	3	Moyenne	1	2	3	Moyenne	1	2	3	Moyenne
IDF1	62.5	49.8	56.8	57.4	72.6	56.5	58.7	65.2	70.6	53.2	54.8	62.6	75.4	53.3	55.4	65.2
FP	189	347	16	552	191	347	15	553	182	346	14	542	183	348	12	543
FN	406	274	55	735	413	275	58	746	417	274	61	752	408	274	62	744
IDSW	30	19	28	77	16	14	28	58	23	19	20	62	11	13	21	45
AssA	42.06				54.9				51.19				54.43			
Frag	40.9				52.94				50.9				53.58			

FIGURE 4.16 – Variations de x

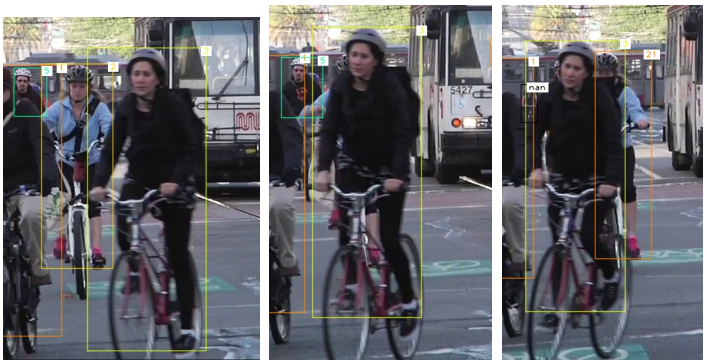
Le facteur x permet de faire varier la contribution qu'ont les distances spatio-temporelles dans la fonction de coût totale. Plus x est grand, moins celles-ci auront d'importance. Globalement, une petite valeur de x provoque des résultats moins bons. C'est expliquable par le fait que le modèle de mouvement des détections dans les images est très approximatif (il ne tient pas compte de l'accélération par exemple). Par conséquent, les valeurs de g_{sp} sont moins fiables, et il est bon de ne pas trop leurs donner d'importance.

τ_{max} :

τ_{max}	2				6				10				14			
Vidéo	1	2	3	Moyenne	1	2	3	Moyenne	1	2	3	Moyenne	1	2	3	Moyenne
IDF1	71.2	50.1	52.0	61.5	70.6	51.8	54.8	62.1	70.6	53.2	54.8	62.6	70.6	52.1	54.8	62.2
FP	180	335	13	528	185	344	14	543	182	346	14	542	183	346	14	543
FN	415	277	63	755	417	275	61	753	417	274	61	752	417	276	61	543
IDSW	26	20	21	67	24	20	20	64	23	19	20	62	23	17	20	60
AssA	50.95				51.53				51.19				51.82			
Frag	49.97				50.6				50.9				50.85			

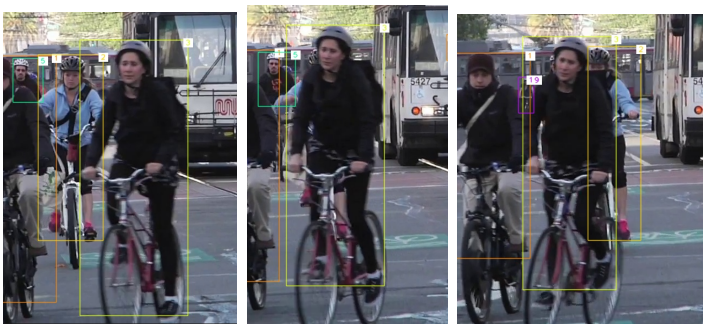
FIGURE 4.17 – Variations de τ_{max}

La valeur τ_{max} détermine la valeur maximale temporelle pour laquelle deux détections peuvent être agrégées. Intuitivement, augmenter τ_{max} permet de résoudre des occlusions de plus en plus grande. Un désavantage notable sera qu'il y aura plus de lien entre tous les noeuds, donc plus d'IDSW potentiels, et aussi plus de temps de calcul. Ce n'est pas très visible dans le tableau 4.17, car les deux effets produisent des IDSW. Cependant, l'exemple des figures 4.18 et 4.19 montrent que pour un τ_{max} plus élevé, la cycliste en bleue peut être correctement identifiée après occlusion.



(a) image 3 *(b) image 11* *(c) image 18*

FIGURE 4.18 – Images extraites de la 2e vidéo : $\tau_{max} = 2$



(a) image 3 *(b) image 11* *(c) image 18*

FIGURE 4.19 – Images extraites de la 2e vidéo : $\tau_{max} = 12$

κ :

Le facteur κ permet de modifier la taille de la fenêtre d'investiguation du plus court chemin, et par conséquent, la taille de δ. A priori, augmenter κ permet de résoudre de plus grandes occlusions, mais augmente aussi la chance d'IDSW.

Numéro de vidéo	1	2	3	Moyenne
IDF1	66.0	53.7	50.5	59.8
FP	190	339	12	541
FN	410	273	70	753
IDSW	19	15	23	57

Table 4.1 – Résultats pour le set de paramètres optimaux : $D_{neutre} = 0.29$, $\alpha_{gsp} = 50$, $x = 2$, $\tau_{max} = 10$, $\kappa = 8$

Néanmoins, cet effet n'est pas particulièrement visible dans la table 4.20 ni dans les images récoltées.

κ	1				2				4				8			
Vidéo	1	2	3	Moyenne	1	2	3	Moyenne	1	2	3	Moyenne	1	2	3	Moyenne
IDF1	68.6	58.6	55.8	63.5	74.9	56.8	55.4	66.2	66.3	58.7	53.8	61.8	76.3	55.0	55.8	66.3
FP	185	343	14	542	183	345	15	543	188	345	13	546	186	350	12	548
FN	418	273	59	750	409	272	58	739	409	273	58	740	409	277	57	743
IDSW	20	13	16	49	14	16	21	51	26	14	17	57	12	20	17	49
AssA	53.77				56.68				50.39				57.47			
Frag	52.21				55.91				48.48				55.91			

Figure 4.20 – Variations de κ

Test finaux

Nous choisissons les paramètres qui donnent de meilleurs résultats globaux, et nous testons ceux-ci sur le set de validations. Comme attendu, les résultats sont médiocres, sur la table 4.1. Les paramètres ne sont définitvement pas indépendants entre eux, ce qui infirme l'hypothèse que nous avons posé plus tôt.

En outre, diminuer la valeur de w^* permet d'acquérir résultats un peu plus prometteurs. En effet, à la place de considérer un poids neutre $f(D^{(v)}_{key,th} + g_{sp,th}) = 0.5$, il semble plus judicieux de revoir à la baisse ce seuil. En utilisant les mêmes hyper paramètres qu'au dessus, mais en réduisant w^* à 0.3, on obtient les résultats visibles sur la table 4.2

Numéro de vidéo	1	2	3	Moyenne
IDF1	75.6	55.9	57.8	66.6
FP	183	341	13	537
FN	408	275	66	749
IDSW	11	10	20	41

Table 4.2 – Résultats pour le set de paramètres optimaux avec $w^* = 0.3$

4.3 Discussion

Il reste énormément de pistes pour optimiser les résultats recceuillis dans ce mémoire. Les évaluations des hyper paramètres se sont faites de manière peu rigoureuses, et sur de trop petits set de vidéos. Au final, les évaluations des hyper paramètres n'ont pas été spécialement faites pour trouver les paramètres optimaux, mais plutôt pour vérifier l'intuition de l'utilité qu'a chaque paramètre. Malheureusement, les erreurs dues au détecteur biaisent fortement les résultats des métriques et il nous est parfois dur de tirer des conclusions grâce à elles. Il est donc important d'aller voir les résultats directement sur des exemples concrets, et pour chaque cas difficile, déterminer le ou les valeurs(s) de paramètres qui empêchent d'obtenir l'effet escompte, afin de résoudre le problème.

Au final, les résultats obtenus en utilisant la fonction de coût originale sont légèrement meilleurs que ceux qu'on a en utilisant la version modifiée de la fonction de coût de base. Néanmoins, la fonction de coût originale offre plus de possibilité d'améliorations et optimisations.

Les distances spatio-temporelles auraient pu être plus précises. En effet, le vecteur de position des détections utilisé actuellement est le point se trouvant au centre de l'encadrement des détections. Utiliser les quatre côté du rectangle permettrait d'obtenir plus de précision.

Finalement, à cause de problèmes techniques, je ne suis pas parvenu à trouver le temps d'effectuer toutes les simulations que je voulais. Comme dit précedement, une méthode comme la recherche par grille m'aurait fort aidé dans la recherche des points optimaux des paramètres. Je n'ai pas pu non plus tester le modèle final sur le set de test. Cependant, comme mes paramètres ne sont pas optimisés, faire cela n'aurait pas apporté grand chose de plus.

Conclusion

Cette thèse a été écrite dans le but d'étudier le suivi d'objets par calculs d'affinités dans un graphe à l'aide de descripteurs d'apparence obtenus par réseaux de neurones. Le chapitre 1 nous a permis de mettre en place les notions nécessaires à la compréhension du milieu du suivi d'objets. Ensuite, le but du chapitre 2 fut de détailler au maximum le ré-identificateur BpbReid, qui est actuellement le seul capable de sa catégorie à correctement comparer des identités de manière globale et locale, rigoureusement. Nous avons pu aussi voir comment fonctionne l'IHT. Ce chapitre nous a permis de pouvoir saisir concrètement la question de ce mémoire : est-ce possible d'effectuer un suivi de personnes en utilisant un algorithme basé sur un graphe a l'aide de descripteurs locaux, dans un contexte sportif où il est fréquent que des cibles soient partiellement voir totalement cachées ? Le chapitre 3 explore la méthodologie adoptée pour élaborer la pondération des arêtes du graphe des détections. Enfin, nous avons pu observer les résultats experimentaux découlant directement de la théorie, et voir dans quels cas l'association se faisait correctement.

Après tout ce travail, il est légitime de se demander si nous sommes en mesure de répondre à la question originelle de ce mémoire. En vérité, il m'est difficile de donner une réponse. Comme dit dans la section 1, les erreurs de localisation sont provoquées par le détecteur. Malheureusement, ces erreurs impactent directement sur les performances du ré-identificateur et surtout de l'algorithme d'association. C'est d'autant plus dérangeant que ces erreurs peuvent engendrer des résultats contre intuitifs si nous nous basons uniquement sur les métriques.

Malgré cela, le tracker final obtenu permet d'en général correctement suivre des personnes dans des situations parfois compliquées (occlusions, mouvement brusques, flous de caméra, détections mal encadrées) et ce, en dépit des métriques peu concluantes. Je regrette de ne pas avoir pu plus tester les différentes combinaisons d'hyper-paramètres.

En conclusion, j'ai le sentiment que l'IHT peut donner de très bon résultats, combiné avec le BpbReid. Il faudrait au préalable que le détecteur puisse mieux

fonctionner dans les situations où les détections sont partiellement cachées, ou dans des poses compliquées. Le détecteur a aussi un rôle crucial dans cette tâche.

Bibliographie

[1] C Amit Kumar K, Damien Delannay, and Christophe De Vleeschouwer. Iterative hypothesis testing for multi-object tracking in presence of features with variable reliability. *arXiv e-prints*, pages arXiv–1509, 2015.

[2] Yaakov Bar-Shalom, Thomas E Fortmann, and Peter G Cable. Tracking and data association, 1990.

[3] Jerome Berclaz, Francois Fleuret, Engin Turetken, and Pascal Fua. Multiple object tracking using k-shortest paths optimization. *IEEE transactions on pattern analysis and machine intelligence*, 33(9) :1806–1819, 2011.

[4] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance : the clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008 :1–10, 2008.

[5] Patrick Dendorfer, Aljosa Osep, Anton Milan, Konrad Schindler, Daniel Cremers, Ian Reid, Stefan Roth, and Laura Leal-Taixé. Motchallenge : A benchmark for single-camera multiple target tracking. *International Journal of Computer Vision*, 129 :845–881, 2021.

[6] Andreas Doering, Di Chen, Shanshan Zhang, Bernt Schiele, and Juergen Gall. Posetrack21 : A dataset for person search, multi-object tracking and multi-person pose tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20963–20972, 2022.

[7] Xingping Dong and Jianbing Shen. Triplet loss in siamese network for object tracking. In *Proceedings of the European conference on computer vision (ECCV)*, pages 459–474, 2018.

[8] Steven Gold, Anand Rangarajan, et al. Softmax to softassign : Neural network algorithms for combinatorial optimization. *Journal of Artificial Neural Networks*, 2(4) :381–399, 1996.

[9] Chang Huang, Bo Wu, and Ramakant Nevatia. Robust object tracking by hierarchical association of detection responses. In *Computer Vision–ECCV 2008 : 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part II 10*, pages 788–801. Springer, 2008.

[10] Zeming Li, Chao Peng, Gang Yu, Xiangyu Zhang, Yangdong Deng, and Jian Sun. Detnet : A backbone network for object detection. *arXiv preprint arXiv :1804.06215*, 2018.

[11] Jonathon Luiten, Aljosa Osep, Patrick Dendorfer, Philip Torr, Andreas Geiger, Laura Leal-Taixé, and Bastian Leibe. Hota : A higher order metric for evaluating multi-object tracking. *International journal of computer vision*, 129 :548–578, 2021.

[12] Hao Luo, Youzhi Gu, Xingyu Liao, Shenqi Lai, and Wei Jiang. Bag of tricks and a strong baseline for deep person re-identification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 0–0, 2019.

[13] Yunjie Peng, Saihui Hou, Chunshui Cao, Xu Liu, Yongzhen Huang, and Zhiqiang He. Deep learning-based occluded person re-identification : A survey. *arXiv preprint arXiv :2207.14452*, 2022.

[14] Md Atiqur Rahman and Yang Wang. Optimizing intersection-over-union in deep neural networks for image segmentation. In *International symposium on visual computing*, pages 234–244. Springer, 2016.

[15] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *European conference on computer vision*, pages 17–35. Springer, 2016.

[16] Vladimir Somers, Christophe De Vleeschouwer, and Alexandre Alahi. Body Part-Based Representation Learning for Occluded Person Re-Identification. *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV23)*, nov 2023.

[17] Yifan Sun, Liang Zheng, Yi Yang, Qi Tian, and Shengjin Wang. Beyond part models : Person retrieval with refined part pooling (and a strong convolutional baseline). In *Proceedings of the European conference on computer vision (ECCV)*, pages 480–496, 2018.

[18] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[19] Mang Ye, Jianbing Shen, Gaojie Lin, Tao Xiang, Ling Shao, and Steven CH Hoi. Deep learning for person re-identification : A survey and outlook. *IEEE transactions on pattern analysis and machine intelligence*, 44(6) :2872–2893, 2021.

[20] Li Zhang, Yuan Li, and Ramakant Nevatia. Global data association for multi-object tracking using network flows. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2008.

[21] Zhong Zhang, Haijia Zhang, and Shuang Liu. Person re-identification using heterogeneous local graph attention networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12136–12145, 2021.

[22] Kuan Zhu, Haiyun Guo, Zhiwei Liu, Ming Tang, and Jinqiao Wang. Identity-guided human semantic parsing for person re-identification. In *Computer Vision–ECCV 2020 : 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 346–363. Springer, 2020.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**