Challenge mathématiques et entreprises : Eurecam

Olympio Hacquard, Etienne Lasalle, Vadim Lebovici

Challenge Mathématiques et Entreprises
Sujet : Eurecam

Olympio Hacquard, Etienne Lasalle, Vadim Lebovici

September 2021

Table des matières

Introduction

Le problème posé par la société Eurecam est celui de la reconstruction de trajectoires de personnes, à partir de leur détection par caméra 3D. Mathématiquement, pour une vidéo consistant en une séquence d'images $v_1, \dots, v_k$, on dispose pour chaque image v_i des coordonnées de points $p_1^{(v_i)}, \dots, p_n^{(v_i)}$ de $\mathbb{R}^3$. Ces coordonnées, au format (x, y, h), représentent respectivement l'abscisse, l'ordonnée et la hauteur des détections. On cherche alors à reconstruire les trajectoires des personnes visibles, i.e. à reconstruire des objets de la forme

$$\left(p_{j_1}^{(v_{i_1})}, p_{j_2}^{(v_{i_2})}, \dots, p_{j_k}^{(v_{i_k})}\right),$$

où chacun des points d'une trajectoire correspond à la même personne sur une image différente. Le problème revient donc à construire des appariements successifs. Les obstacles observés en pratique sont les suivants :

— certaines données sont aberrantes en raison du bruit sur l'image et ne correspondent pas à l'observation d'une personne,
— le nombre d'observations pour une personne donnée n'est pas constant au fil des images,
— certaines données sont manquantes, de sorte qu'une personne peut n'avoir aucune observation sur une image et réapparaître à l'image suivante.

Ces deux premiers obstacles imposent un pré-traitement des données, présenté en section 1. D'une part, il vise à filtrer les détections aberrantes et d'autre part, il remplace les détections restantes par des données calculées n'ayant qu'un unique point par personne. Afin d'effectuer un appariement des points entre deux images successives, nous avons choisi d'utiliser un modèle de transport optimal, exposé en section 2. Nous avons adapté l'algorithme de transport optimal classique afin de prendre en compte les variations du nombre de points entre deux images consécutives induites par les entrées et sorties de personnes. De plus, nous proposons une méthode de transport optimal permettant une certaine robustesse aux données manquantes. La section 3 présente quelques étapes de post-traitement afin de s'assurer que l'absence de données n'a pas coupé de trajectoire en deux et de nettoyer les trajectoires obtenues. Enfin, un index décrit les paramètres de notre modèle en appendice.

Le code est disponible en ligne : `https://github.com/elasalle/challengeMathEntreprises`. Pour des raisons de confidentialité, les données sont réduites aux détections sans les vidéos correspondantes.

1 Pré-traitement

La partie de pré-traitement des données vise à obtenir un nuage de points ne contenant qu'un unique point par personne physique à partir duquel la reconstruction de trajectoires sera faite.

Procédure. Partant des données de détections du fichier `detection.txt`, on applique la procédure suivante :

1. Les détections aberrantes sont supprimées au préalable si :
   (a) leur hauteur est inférieure ou égale à un seuil $h_{\min}$ choisi *a priori* et correspondant à `height_thresh` de `best_parameters.csv`,
   (b) elles se trouvent en dehors d'un domaine géométrique fixé. Nous avons pris le rectangle $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$ défini par les paramètres `x_min`, `x_max`, `y_min`, `y_max` de `best_parameters.csv`, mais il est tout à fait possible d'implémenter des domaines plus complexes, non convexes par exemple.
2. Le nuage de points final correspond alors à l'ensemble des *maxima locaux* de la fonction $f : \mathbb{R}^2 \to \mathbb{R}$ construite en sommant les gaussiennes

$$f_i : \begin{array}{ccc} \mathbb{R}^2 & \to & \mathbb{R} \\ (x, y) & \mapsto & h_i \cdot \exp\left(-\frac{(x-x_i)^2+(y-y_i)^2}{2\sigma^2}\right) \end{array}$$

pour chaque point $p_i = (x_i, y_i, h_i)$ du nuage obtenu après l'étape 1. De plus, chaque maximum de f se voit assigner la hauteur du point p_i le plus proche de lui pour permettre l'utilisation de l'information de hauteur lors de la reconstitution des trajectoires.

Nous choisissons h_i comme amplitude pour f_i afin de diminuer l'influence de mauvaises détections non filtrées à l'étape 1 (mains, sac à dos, ...) dont les hauteurs sont souvent plus faibles.

L'écart-type σ est choisi *a priori* et correspond à `sigma` de `best_parameters.csv`. Il est de l'ordre d'une dizaine de centimètres pour correspondre au rayon horizontal typique d'une personne.

Implémentation. La fonction `get_clean_detections` permettant le pré-traitement des données selon la procédure précédente est implémentée et décrite dans le fichier `tools_for_preprocessing.py`. La calcul des maxima locaux de f est effectué via la fonction `peak_local_maxima` du module `feature` de la librairie scikit-image [Van+14].

Illustration. L'étape 1 – le filtrage des données aberrantes – est bien illustrée par la figure 1. Le filtrage géométrique permet de supprimer les points provenant de la détection des poubelles et du mur, tandis que le filtrage par la hauteur permet de supprimer les détections des membres des personnes. Ces deux étapes augmentent la précision du calcul de la position des personnes à l'étape 2 du pré-traitement.

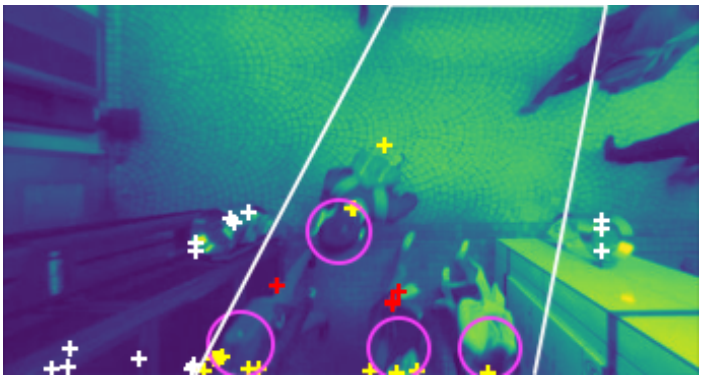


FIGURE 1 – Exemple de filtrage des données aberrantes sur l'image 570 du jeu de données n°4. Pour une meilleure lisibilité, nous avons entouré les personnes en rose. Le trapèze blanc est la projection au sol du rectangle $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$. Les points rouges sont les détections filtrées par leur hauteur, les blancs celles filtrées par leur position et les points jaunes les détections conservées.

L'étape 2 – le calcul du nuage de point final – est illustrée sur la figure 1. On peut constater que les maxima de f correspondent bien chacun à une personne sur l'image, exceptée la personne la plus à droite en figure 1 dont la gaussienne ne suffit pas à créer un maximum pour f. Ce sera cependant le cas dans les images suivantes.

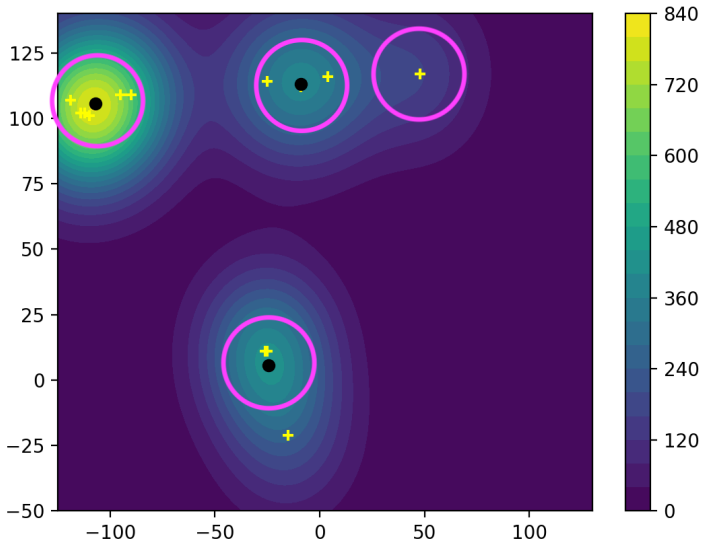


FIGURE 2 – Illustration de la construction du nuage de points final sur l'image 570 du jeu de données n°4 avec $\sigma = 21$. Pour une meilleure lisibilité, nous avons entouré les personnes en rose. Les croix jaunes représentent les projections sur le plan (x, y) des détections $p_i = (x_i, y_i, h_i)$ conservées après le premier filtrage. Les niveaux de f sont représentés en couleur sur le plan. Le nuage de points final est formé par les trois maxima locaux de f, en noir sur la figure. L'axe y est inversé par rapport à la figure 1.

2 Reconstruction des trajectoires

A partir des données issues du pré-traitement, nous avons choisi de reconstruire les trajectoires, transition par transition, en effectuant un appariement des points d'une image à ceux de la suivante. Cet appariement est obtenu en résolvant un problème de transport optimal avec bord, permettant ainsi de gérer les apparitions et disparitions de personnes. Lorsqu'une personne arrive du bord, elle démarre une trajectoire. Cette dernière est mise à jour au fur et à mesure des transitions, et se termine lorsqu'un point est envoyé sur le bord. Nous avons proposé un coût de transport prenant en compte à la fois les positions des personnes mais également leurs tailles. De plus, pour prendre en compte les données manquantes (personnes non détectées sur quelques images), on propose une modification du problème de transport et des coûts, pour autoriser les personnes à rester stationnaires pendant un certain temps.

2.1 Transport optimal avec bords

Cadre théorique. Dans le cadre général du transport optimal avec bord [FG10], on considère deux mesures μ et ν supportées sur un domaine $\mathcal{X}$ avec un bord $\partial\mathcal{X}$. On notera $\overline{\mathcal{X}} = \mathcal{X} \sqcup \partial\mathcal{X}$, l'union disjointe du domaine et de son bord. On considérera une distance d sur $\overline{\mathcal{X}}$. On appelle *plan de transport admissible*, une mesure π sur $\overline{\mathcal{X}} \times \overline{\mathcal{X}}$ vérifiant les contraintes marginales suivantes : pour tout borélien $A \subset \mathcal{X}$,

$$\pi(A \times \overline{\mathcal{X}}) = \mu(A) \quad \text{et} \quad \pi(\overline{\mathcal{X}} \times A) = \nu(A). \tag{1}$$

Le coût total d'un tel plan de transport est donné par

$$\iint_{\overline{\mathcal{X}}\times\overline{\mathcal{X}}} d(x,y)^2 d\pi(x,y). \tag{2}$$

On appellera *plan de transport optimal*, un plan de transport admissible minimisant le coût total. Moralement, on s'autorise à déplacer un élément de masse $d\mu(x)$ sur $d\nu(y)$ avec un coût $d(x,y)^2$, ou vers sa projection sur le bord $s(x)$ avec un coût $d(x,s(x))^2$. Il est à noter qu'ici le coût d'un déplacement correspond au carré de la distance. Par la suite, nous considérerons des coûts plus adaptés à notre problème.

Application. Nous allons appliquer cette approche pour apparier les personnes entre deux images successives. Soient $p_1, \dots, p_n$ les positions des personnes d'une image et $p'_1, \dots, p'_{n'}$ celles des personnes de l'image suivante. On considère les mesures discrètes $\mu = \sum \delta_{p_i}$ et $\nu = \sum \delta_{p'_j}$. Ainsi, résoudre le problème de transport optimal avec bord revient à trouver un appariement partiel entres les (p_i) et les (p'_j), en autorisant les points à être envoyés sur le bord de l'image. Pour résoudre ce problème de transport optimal avec bord en pratique, on résout un problème de transport optimal classique, en rajoutant le bord aux points de départ et aux points d'arrivée. On cherche donc un plan de transport optimal entre les mesures $\hat{\mu}$ et $\hat{\nu}$ définies par les vecteurs

$$\hat{\mu} = (\underbrace{1, \dots, 1}_{n}, n') \quad \text{et} \quad \hat{\nu} = (\underbrace{1, \dots, 1}_{n'}, n),$$

où chaque personne possède une masse 1, tandis que le bord a soit une masse n' pour $\hat{\mu}$, soit une masse n pour $\hat{\nu}$. Les deux mesures sont donc de masse totale égale à $n + n'$. Cette condition d'égalité des masses totales est indispensable pour pouvoir effectuer du transport optimal standard. On décompose ensuite la matrice des coûts comme suit :

$$\mathcal{C} = \left(\begin{array}{c|c} C & a \\ \hline b^T & 0 \end{array}\right).$$

La matrice C, de taille $n \times n'$ représente le coup pour appairer les personnes, *i.e.* $C_{i,j}$ représente le coût pour appairer p_i avec p'_j. De même, a_i représente le coût de transport de p_i sur le bord, et b_j

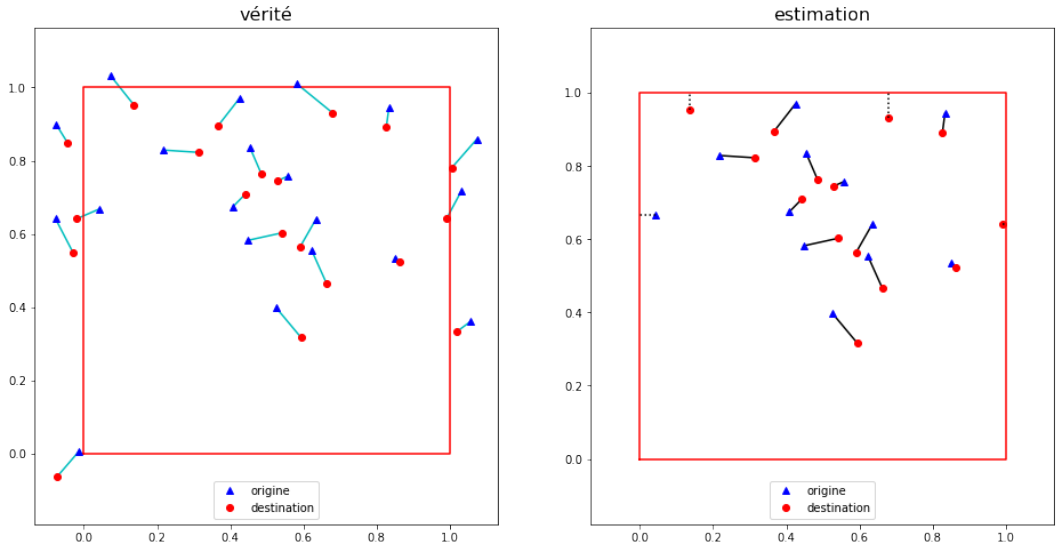


FIGURE 3 – Exemple d'appariement de personnes. Le carré rouge représente le domaine considéré. Les points bleus représentent la position initiale de personnes, les points rouges leur position finale. A gauche, les données brutes. A droite, une reconstruction des trajectoires à l'aide du transport optimal avec bord, en n'utilisant que les personnes à l'intérieur du domaine.

représente le coût de transport d'une masse 1 du bord vers p'_j. Le coefficient zéro indique un coût nul pour envoyer de la masse du bord vers le bord.

Résoudre ce problème de transport optimal, revient à trouver la matrice P du plan de transport, c'est à dire une matrice à coefficients positifs, de taille $(n+1) \times (n'+1)$, qui vérifie $P1_{n'+1} = \hat{\mu}^T$ et $1_{n+1}^T P = \hat{\nu}$, avec 1_k le vecteur de taille k où toutes les coordonnées sont égales à 1, et qui minimise le coût de transport total

$$\sum_{i=1}^{n+1} \sum_{j=1}^{n'+1} \mathcal{C}_{i,j} P_{i,j}.$$

Les appariements sont alors donnés par les coefficients non nuls de la matrice P. En résumé, le transport optimal considère tous les couplages possibles entre deux images successives et renvoie celui qui minimise la somme des coûts élémentaires associés à chaque déplacement.

La construction de la matrice des coûts pour effectuer le transport optimal avec bord est inspirée de l'implémentation de la fonction `wasserstein_distance` par [LCO18] dans la libraire Gudhi [Mar+14]. Dans notre programme, les calculs de transport optimal sont effectués en utilisant la libraire POT : Python Optimal Transport [Fla+21].

Illustration La figure 3 montre un exemple d'appariement de personnes sur un jeu de données synthétique. Cet appariement est calculé en prenant comme coût les carrés des distances euclidiennes (soit entre les personnes, soit entres les personnes et le bord du domaine). On constate sur cet exemple que cette approche permet de détecter les arrivées de personnes (voir le bord haut du domaine) ainsi que les départs (voir le bord gauche du domaine).

2.2 Choix du coût

Pour réaliser le transport optimal des personnes d'une image à la suivante, on choisit d'utiliser les coordonnées $p = (x, y, h)$ des personnes. Cependant, plutôt que d'utiliser la distance euclidienne standard pour définir le coût de transport, on introduit un paramètre λ, qui réalise un pénalisation par la hauteur des personnes. Le coût pour envoyer un point $p_i = (x_i, y_i, h_i)$ sur $p'_j = (x'_j, y'_j, h'_j)$ est

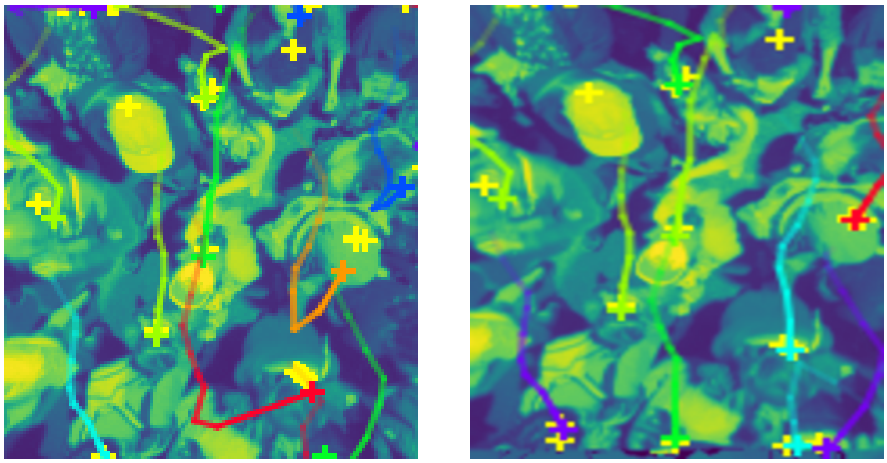
alors

$$C_{i,j} = (x_i - x'_j)^2 + (y_i - y'_j)^2 + \lambda(h_i - h'_j)^2. \tag{3}$$

En prenant $\lambda > 1$, on pénalise d'avantage le fait d'apparier des points ayant des hauteurs différentes. Dans notre programme, ce coefficient λ est défini via le paramètre `pen` de `best_parameters.csv`.

Pour les coefficients a_i et b_j, on utilise cette fois la distance euclidienne entre les personnes et le bord du domaine, sans considérer la hauteur. On peut cependant raffiner la définition du domaine, en considérant par exemple uniquement les distances aux bords haut et bas de l'image, s'il est impossible pour les personnes de sortir sur les côtés.

Illustration. On peut observer figure 4 l'effet de la pénalisation sur le jeu de données n°6 où l'on observe une foule avançant à vitesse constante. Si l'on ne pénalise pas la hauteur, un mauvais appariement a eu lieu causant une propagation d'erreur qui a fait que chaque personne s'est retrouvée appariée avec la personne juste avant elle dans la foule. Si l'on pénalise la hauteur, on empêche ce mauvais appariement de se faire, et les trajectoires sont bien conservées sur la même personne. On note qu'en pratique, pénaliser la hauteur ne fournit pas forcément une reconstitution plus fidèle. En particulier, si les mesures de hauteur d'une personne sont entachées d'erreurs, on veut prendre garde à ne pas amplifier ces erreurs.



(a) $\lambda = 1$ (b) $\lambda = 3$

FIGURE 4 – Effet de la pénalisation sur les trajectoires

2.3 Stationnement

Dans certains jeux de données, on constate que des personnes ne sont pas reconnues durant quelques images, avant d'être de nouveau repérées en raison soit de données manquantes, soit d'un choix d'écart-type trop large pour le partitionnement en pré-traitement qui résulte en la fusion de deux personnes. Dans une telle situation, le transport optimal avec bord va envoyer certaines personnes sur le bord à tort, ou apparier les mauvaises personnes. En pratique, on observe des déplacements beaucoup trop rapides de personnes que l'on considère comme irréalistes compte tenu de la vitesse maximale d'une personne. Pour pallier à ce problème, on choisit d'autoriser les personnes à rester stationnaires un certain temps et de pénaliser les déplacements de longueurs supérieures à un seuil (voir figure 6).

Pour cela, on se fixe des variables $s_{\max} \in \mathbb{N}^*$, $\alpha \geq 1$ et $d_{\max} > 0$, correspondant respectivement aux paramètres `max_statio`, `statio_slope` et `statio_thresh` définie dans `best_parameters.csv`. Considérons comme précédemment les points $(p_i)_{1\leq i\leq n}$ et $(p'_j)_{1\leq j\leq n'}$ de deux images successives. On définit $s \in (\mathbb{N}^*)^n$, où s_i est le nombre d'images pendant lesquelles la personne était au point p_i. Par exemple, si la personne n'a encore jamais patienté à ce point, $s_i = 1$.

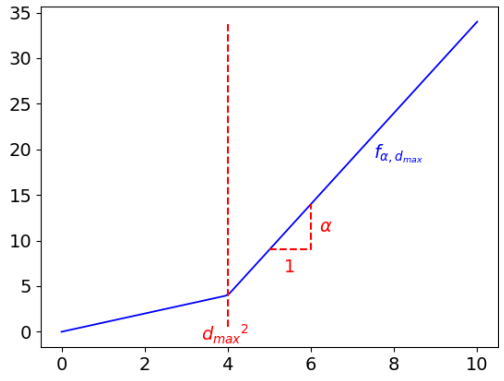


FIGURE 5 – Graphe de la fonction qui pénalise les grands déplacement : $f_{\alpha,d_{\max}}$.

Pour pénaliser les déplacement trop grands, on remplace le coût $C_{i,j}$ défini en (3) par $\tilde{C}_{i,j} = f_{\alpha,s_i d_{\max}}(C_{i,j})$ où, $f_{\alpha,d}$ est la fonction continue sur $\mathbb{R}_+$, affine par morceaux, égale à l'identité sur $[0, d^2]$ et de pente α sur $[d^2, +\infty[$ (voir figure 5). De même a_i est remplacé par $\tilde{a}_i = f_{\alpha,s_i d_{\max}}(a_i)$ et b_j est remplacé par $\tilde{b}_j = f_{\alpha,d_{\max}}(b_j)$.

Pour autoriser les personnes à rester stationnaires, on rajoute les points (p_i) aux points de l'image suivante, en étendant la matrice des coûts comme suit :

$$\tilde{\mathcal{C}} = \left(\begin{array}{c|ccc|c} \multicolumn{3}{c|}{\multirow{2}{*}{$\tilde{C}$}} & & \multirow{2}{*}{$\tilde{a}$} \\ & & \tilde{D} & & \\ \hline \tilde{b}^T & 0 & \cdots & 0 & 0 \end{array}\right) \quad \text{où} \quad \tilde{D} = \begin{pmatrix} \tilde{D}_{1,1} & +\infty & \cdots & +\infty \\ +\infty & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & +\infty \\ +\infty & \cdots & +\infty & \tilde{D}_{n,n} \end{pmatrix},$$

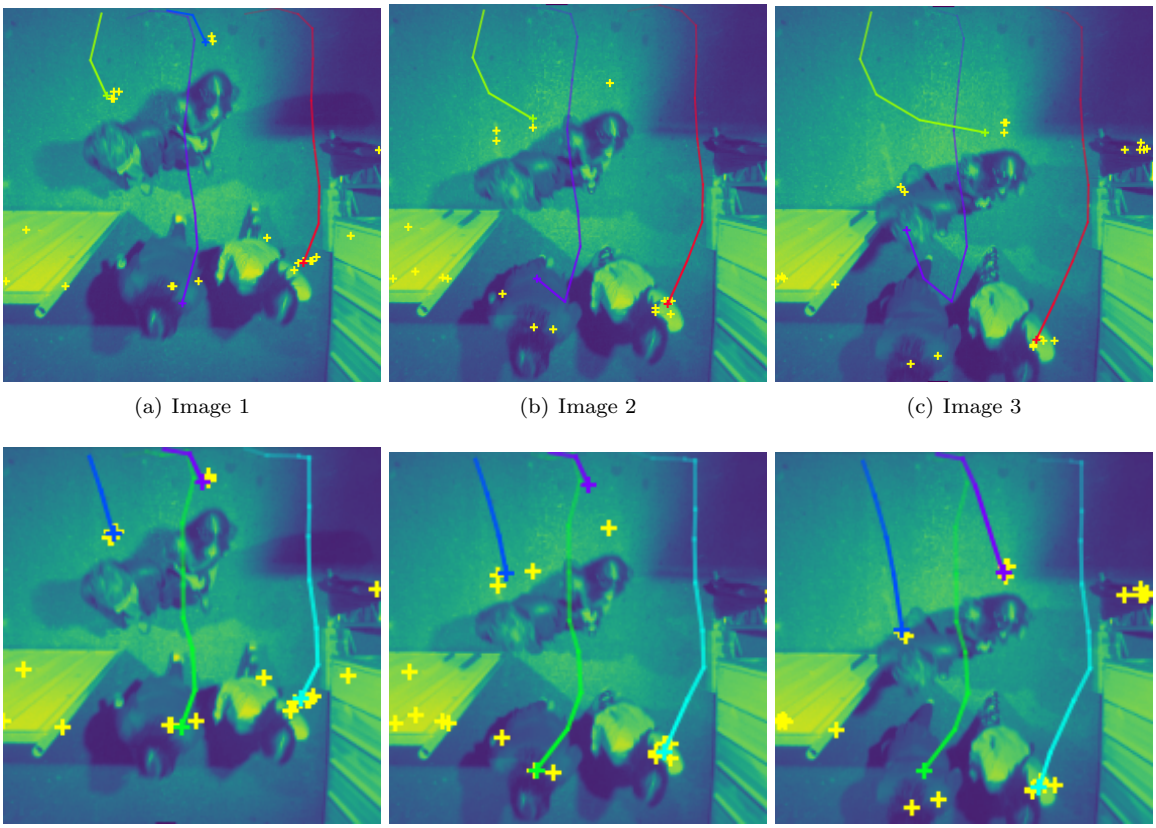
avec

$$\tilde{D}_{i,i} = \begin{cases} +\infty & \text{si } s_i \geq s_{\max}; \\ (s_i d_{\max})^2 & \text{sinon.} \end{cases}$$

Les coefficients diagonaux de $\tilde{D}$ correspondent aux coûts pour qu'une personne reste stationnaire. Le choix de ce coût provient du fait que $d_{\max}$ correspond moralement à la distance maximale autorisée lors d'un déplacement en une transition. Ainsi, si la personne est restée stationnaires pendant s_i transitions, on lui autorise un déplacement coûtant $(s_i d_{\max})^2$. Les $+\infty$ apparaissant sur les coefficients hors de la diagonale permettent d'interdire l'appariement entre les points de l'image de départ.

Implémentation. La fonction `OT_step` du fichier `tools_for_reconstruction.py` code le calcul d'un plan de transport optimal en fonction des coordonnées des détections et de celles du domaine. Il est possible d'autoriser ou d'interdire le stationnement via le paramètre `allow_statio`. `OT_step` est utilisée de manière itérative par la fonction `get_trajectories_from_detections` du même fichier, pour reconstruire les trajectoires.

Illustration. On peut voir en figure 6, issue du jeu de données n°2, l'intérêt du stationnement. Sur la bande du haut (sans stationnement), il y a une donnée manquante à la deuxième image. En conséquence, la trajectoire verte de la personne du haut va venir s'apparier avec la mauvaise personne. Cela entraîne une propagation d'erreur car la trajectoire violette va venir s'apparier sur la personne du haut. Sur la bande du bas où la stationnement est autorisé, la trajectoire violette stagne à la deuxième image où elle n'a pas de donnée correspondante et reprend sa course à la troisième image. Les quatre trajectoires sont donc fidèlement reconstruites.



(a) Image 1 (b) Image 2 (c) Image 3

(d) Image 1 (e) Image 2 (f) Image 3

FIGURE 6 – En haut : trajectoires avec stationnement interdit. En bas : trajectoires avec stationnement possible.

3 Post-traitement

Procédure.

1. Lorsqu'une donnée manque pour le suivi d'une personne, cela résulte en deux trajectoires distinctes correspondant pourtant à la même personne. Pour corriger ce problème, nous recollons les trajectoires ayant une fin trop loin du bord à celles ayant un début trop loin du bord. Le recollement ne se fait que si la fin de la première est suffisamment proche du début de la deuxième *spatialement et temporellement*.
   Plus précisément, pour une trajectoire se terminant à l'image u en un point p tel que $d(p, \partial\mathcal{X}) > d_{\max}$, on cherche une trajectoire naissant à une image $v \geq u$ et en un point p' tel que $d(p', \partial\mathcal{X}) > d_{\max}$ et telle que
   $$\|p - p'\| \leq (v - u) d_{\max},$$
   $$v - u \leq s_{\max}.$$
   Si une telle trajectoire existe, on la concatène à la première et si plusieurs recollements sont possibles, on en choisit un arbitrairement.
2. Les trajectoires courtes correspondant le plus souvent à du bruit résiduel en pratique, nous éliminons celles ayant une longueur inférieure à un seuil $l_{\min}$ correspondant à `min_size` dans `best_parameters.csv`.

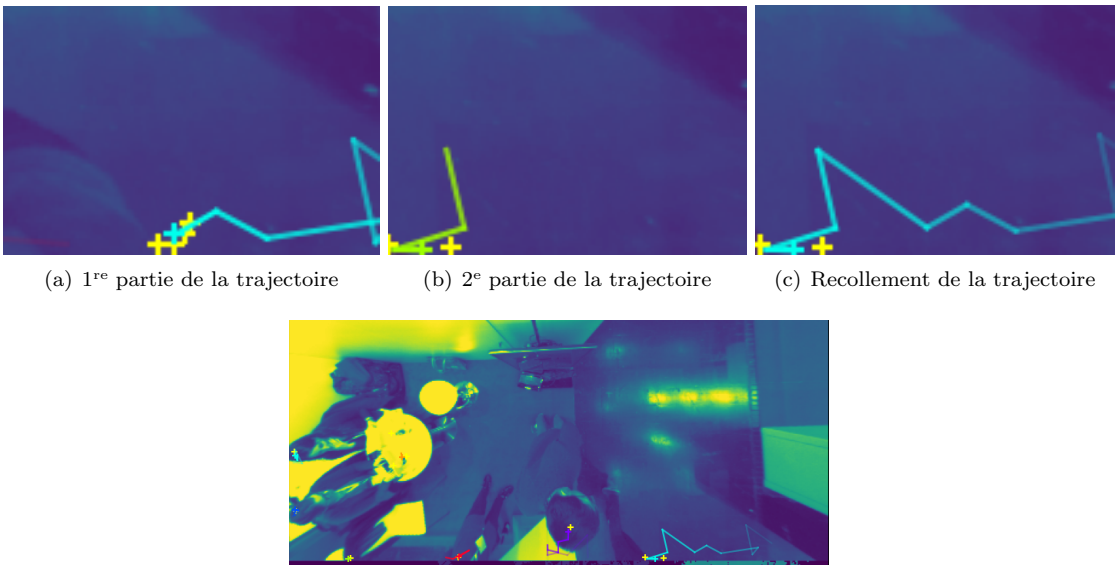
Nous avons appliqué cette méthode de post-traitement *après* la collecte de toutes les trajectoires sur un jeu de données. Néanmoins, cette méthode peut s'adapter pour être utilisée en direct sur les

quelques images précédentes et peut ainsi être utilisée quasiment en temps réel pour l'application souhaitée par Eurecam.

Implémentation.

1. La fonction `glue_trajectories` permettant le recollement des trajectoires est implémentée et décrite dans le fichier `tools_for_postprocessing.py`.
2. La fonction `delete_small_trajectories` permettant la suppression des petites trajectoires est implémentée et décrite dans le fichier `tools_for_postprocessing.py`.

Illustration. On peut voir en figure 7 en bas à droite de l'image tirée du jeu de données n°9 un recollement de deux trajectoires correspondant à la même personne.



(a) 1re partie de la trajectoire (b) 2e partie de la trajectoire (c) Recollement de la trajectoire

(d) Scène complète

FIGURE 7 – Recollement de trajectoires

Discussion

La reconstruction des trajectoires obtenue grâce à notre méthode s'est avérée être plutôt fidèle à la réalité et comparable aux résultats fournis par Eurecam pour la plupart des jeux de données. Les seuls jeux de données ayant posé des problèmes sont le n°5 où le bruit vient parasiter quelques trajectoires, et le n°6 où la densité de la foule conduit à la disparition de certaines trajectoires et à la confusion de certaines personnes. Un filtrage du bruit utilisant un bord non convexe permettrait sûrement d'améliorer les résultats pour le jeu de données n°5.

La méthode que nous avons développée possède un certain nombre de paramètres à calibrer. Cependant, certains peuvent être fixés une fois pour toutes selon la géométrie de la scène ($h_{\min}$ ainsi que le domaine $x_{\min}, x_{\max}, y_{\min}, y_{\max}$). Le paramètre $d_{\max}$ est fixé par des considérations physiques, de même pour le paramètre σ qui dépend éventuellement de la densité de la foule si l'on cherche à raffiner le modèle. Il reste ainsi les paramètres $\lambda, s_{\max}$ et α qui paraissent plus délicats à calibrer *a priori*. La donnée des trajectoires réelles sur des données d'entraînement permettrait d'optimiser ces paramètres.

Références

[FG10] Alessio Figalli et Nicola Gigli. “A new transportation distance between non-negative measures, with applications to gradients flows with Dirichlet boundary conditions”. In : *Journal de mathématiques pures et appliquées* 94.2 (2010), p. 107-130 (cf. p. 5).

[Fla+21] Rémi Flamary et al. “Pot: Python optimal transport”. In : *Journal of Machine Learning Research* 22.78 (2021), p. 1-8 (cf. p. 6).

[LCO18] Théo Lacombe, Marco Cuturi et Steve Oudot. “Large scale computation of means and clusters for persistence diagrams using optimal transport”. In : *arXiv preprint arXiv:1805.08331* (2018) (cf. p. 6).

[Mar+14] Clément Maria et al. “The gudhi library: Simplicial complexes and persistent homology”. In : *International congress on mathematical software*. Springer. 2014, p. 167-174 (cf. p. 6).

[Van+14] Stefan Van der Walt et al. “scikit-image: image processing in Python”. In : *PeerJ* 2 (2014), e453 (cf. p. 3).

Index des paramètres

Paramètre	Variable	Valeur par défaut	Explication
`height_threshold`	$h_{\min}$	`5`	Hauteur en deçà (au sens large) de laquelle les détections sont considérées comme aberrantes.
`x_min`, `x_max`, `y_min`, `y_max`	$x_{\min}$, $x_{\max}$, $y_{\min}$, $y_{\max}$	`-200`, `200`, `-200`, `200`	Paramètres définissant le rectangle $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$ en dehors duquel les détections sont considérées comme aberrantes.
`sigma`	σ	`20`	Ecart-type des gaussiennes utilisées dans le prétraitement. Des valeurs entre 0 et 15 augmentent le nombre de points et sont préconisées pour des foules denses. Des valeurs entre 15 et 20 sont préconisées pour les foules peu denses.
`pen`	λ	`1`	Coefficient pénalisant les différences de hauteurs dans le transport optimal. Des grandes valeurs forcent un appariement respectant au mieux les hauteurs.
`max_statio`	$s_{\max}$	`3`	Temps de stationnement maximal d'une trajectoire. Un temps élevé peut créer des recollements abusifs et `max_statio = 1` ne permet aucun stationnement. Une valeur entre 1 et 3 est préconisée.
`statio_thresh`	$d_{\max}$	`35`	Distance maximale au-delà de laquelle le déplacement est d'avantage pénalisé. Un calcul via la vitesse maximale d'une personne ($\sim 5\ km/h$) et le nombre d'images par seconde peut donner une première estimation de ce paramètre.
`statio_slope`	α	`2`	Pente contrôlant l'intensité de la pénalisation d'un déplacement au delà de la distance $d_{\max}$.
`min_size`	$l_{\min}$	`3`	Taille minimale en deçà (au sens large) de laquelle une trajectoire est considérée comme étant du bruit résiduel.