

Rapport de Projet : Suivi d'Objets Multi-Cibles

Machine Learning for Visual Object Tracking (MLVOT)

Emre Ulusoy
`emre.ulusoy@epita.fr`

December 14, 2025

Contents

1 TP 1 : Filtre de Kalman pour le Suivi 2D	2
1.1 Contexte et Objectif	2
1.2 Modèle Mathématique	2
1.2.1 Vecteur d'État	2
1.2.2 Modèle de Mouvement	2
1.2.3 Modèle d'Observation	2
1.2.4 Matrices de Covariance	2
1.3 Algorithme du Filtre de Kalman	3
1.4 Implémentation	3
1.5 Visualisation	3
2 TP 2 : Tracker IoU avec Algorithme Hongrois	3
2.1 Contexte et Objectif	3
2.2 Intersection-over-Union (IoU)	4
2.3 Matrice de Similarité	4
2.4 Algorithme Hongrois	4
2.5 Gestion des Trajectoires	5
2.6 Paramètres	5
2.7 Résultat Visuel	5
3 TP 3 : Intégration Kalman-IoU	5
3.1 Motivation	5
3.2 Architecture du Système	6
3.3 Modification du Pipeline	6
3.4 Représentation Hybride	7
3.5 Résultat Visuel	7
4 TP 4 : Intégration de la Ré-identification (ReID)	7
4.1 Motivation	7
4.2 Extracteur de Features	8
4.2.1 Modèle OSNet	8
4.2.2 Prétraitement des Patches	8
4.3 Métriques de Similarité	8
4.3.1 Similarité Cosinus	8

4.3.2	Similarité Normalisée (Distance Euclidienne)	8
4.4	Score Combiné	8
4.5	Lissage Temporel des Features	9
4.6	Résultat Visuel	9
5	Défis Rencontrés et Solutions	9
5.1	Compatibilité OpenCV	9
5.2	Format des Détections MOT	10
5.3	Dimensions des Bounding Boxes	10
5.4	Initialisation de la Covariance	10
5.5	Patches Invalides pour ReID	10
5.6	Équilibrage IoU/ReID	10
6	Résultats Expérimentaux	11
6.1	Métriques d'Évaluation MOT	11
6.1.1	Formule MOTA	11
6.2	Résultats MOTA sur ADL-Rundle-6	11
6.3	Analyse des Résultats MOTA	11
6.3.1	Pourquoi TP2 obtient le meilleur MOTA ?	11
6.3.2	Pourquoi le ReID diminue le MOTA ?	12
6.3.3	Fragmentation des Trajectoires	12
6.4	Comparaison Temps de Traitement	12
6.5	Visualisation et Vidéos de Sortie	12
7	Conclusion et Perspectives	13
7.1	Bilan	13
7.2	Trade-off MOTA vs Cohérence	13
7.3	Limitations	13
7.4	Améliorations Futures	13

1 TP 1 : Filtre de Kalman pour le Suivi 2D

1.1 Contexte et Objectif

Le premier exercice consiste à implémenter un filtre de Kalman pour suivre un objet unique (SOT - Single Object Tracking) représenté par son centroïde. Le filtre de Kalman est un estimateur récursif optimal qui prédit l'état futur d'un système dynamique linéaire bruité.

1.2 Modèle Mathématique

1.2.1 Vecteur d'État

Le vecteur d'état à 4 dimensions représente la position et la vitesse :

$$\mathbf{x}_k = \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix} \quad (1)$$

1.2.2 Modèle de Mouvement

Le modèle assume un mouvement à vitesse quasi-constante avec accélération comme entrée de contrôle :

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u} + \mathbf{w}_{k-1} \quad (2)$$

Avec les matrices :

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \frac{\Delta t^2}{2} & 0 \\ 0 & \frac{\Delta t^2}{2} \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \quad (3)$$

1.2.3 Modèle d'Observation

Seules les positions sont observables :

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k, \quad \mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad (4)$$

1.2.4 Matrices de Covariance

La matrice de covariance du bruit de processus \mathbf{Q} modélise l'incertitude sur l'accélération :

$$\mathbf{Q} = \sigma_a^2 \begin{bmatrix} \frac{\Delta t^4}{4} & 0 & \frac{\Delta t^3}{2} & 0 \\ 0 & \frac{\Delta t^4}{4} & 0 & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & 0 & \Delta t^2 & 0 \\ 0 & \frac{\Delta t^3}{2} & 0 & \Delta t^2 \end{bmatrix} \quad (5)$$

La matrice de covariance de mesure :

$$\mathbf{R} = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix} \quad (6)$$

1.3 Algorithme du Filtre de Kalman

Algorithm: Kalman Filter

Prediction (Time Update):

1. $\hat{\mathbf{x}}_k^- = \mathbf{A}\hat{\mathbf{x}}_{k-1} + \mathbf{B}\mathbf{u}$
2. $\mathbf{P}_k^- = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^T + \mathbf{Q}$

Correction (Measurement Update):

3. $\mathbf{S}_k = \mathbf{H}\mathbf{P}_k^-\mathbf{H}^T + \mathbf{R}$
4. $\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}^T\mathbf{S}_k^{-1}$ (Kalman Gain)
5. $\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k^-)$
6. $\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_k^-$

1.4 Implémentation

L'implémentation comprend deux fichiers principaux :

- `KalmanFilter.py` : Classe du filtre avec méthodes `predict()` et `update()`
- `objTracking.py` : Script principal intégrant la détection et le suivi

1.5 Visualisation

La visualisation utilise un code couleur :

- **Vert** : Cercle détecté (observation)
- **Bleu** : Position prédicté (avant correction)
- **Rouge** : Position estimée (après correction)
- **Jaune** : Trajectoire historique

Note : Une vidéo de démonstration du suivi est disponible dans le dossier de rendu.

2 TP 2 : Tracker IoU avec Algorithme Hongrois

2.1 Contexte et Objectif

Le deuxième exercice étend le suivi à plusieurs objets simultanés (MOT). La représentation passe du centroïde à la bounding box complète, et l'association détection-trajectoire utilise l'Intersection-over-Union (IoU) comme métrique de similarité.

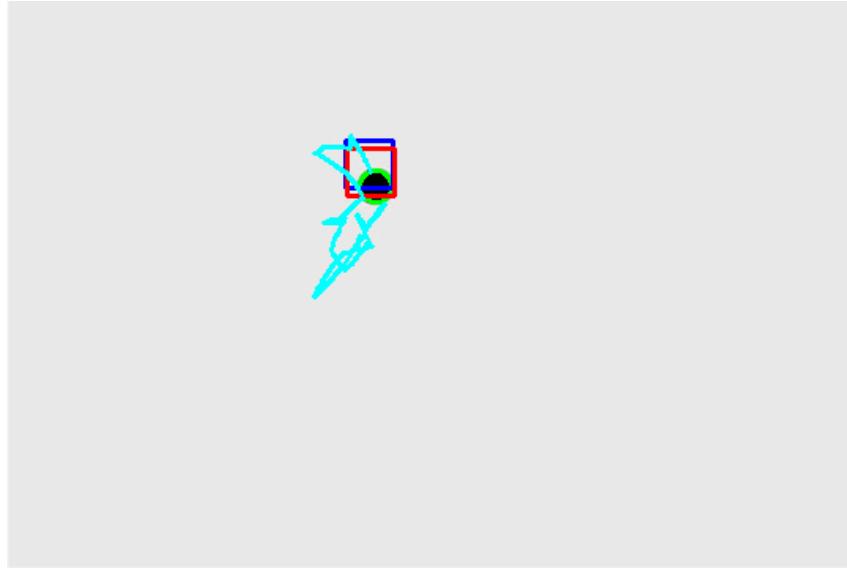


Figure 1: TP1 : Suivi d'une balle avec le filtre de Kalman. Le cercle vert représente la détection, le rectangle bleu la prédiction, et le rectangle rouge l'estimation corrigée. La trajectoire jaune montre le chemin parcouru.

2.2 Intersection-over-Union (IoU)

L'IoU, ou indice de Jaccard, mesure le chevauchement entre deux bounding boxes :

$$\text{IoU}(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{\text{Aire d'intersection}}{\text{Aire d'union}} \quad (7)$$

IoU Illustration
Box A: $[x_1, y_1, w_1, h_1]$
Box B: $[x_2, y_2, w_2, h_2]$
$\text{IoU} = \frac{\text{Area}(A \cap B)}{\text{Area}(A \cup B)}$

2.3 Matrice de Similarité

Pour N trajectoires existantes et M nouvelles détections, on construit une matrice de similarité $\mathbf{S} \in \mathbb{R}^{N \times M}$ où :

$$S_{i,j} = \text{IoU}(\text{Track}_i, \text{Detection}_j) \quad (8)$$

2.4 Algorithme Hongrois

L'association optimale est obtenue par l'algorithme hongrois (Kuhn-Munkres) qui minimise le coût total d'assignation. La matrice de coût est définie comme :

$$C_{i,j} = 1 - S_{i,j} \quad (9)$$

L'algorithme résout le problème d'assignation linéaire en temps $O(n^3)$ et garantit une solution optimale globale.

2.5 Gestion des Trajectoires

Track Management Pipeline

```

Hungarian Algorithm → Matched Tracks → Update Track
                    → Unmatched Detections → Create New Track
                    → Unmatched Tracks → Delete (if  $t > t_{max}$ )

```

Trois cas sont distingués après l'association :

1. **Matched** : La trajectoire est mise à jour avec la nouvelle détection
2. **Unmatched Detections** : Une nouvelle trajectoire est créée
3. **Unmatched Tracks** : La trajectoire est marquée comme manquée et supprimée après t_{max} frames

2.6 Paramètres

Paramètre	Symbol	Valeur
Seuil IoU minimum	τ_{IoU}	0.3
Frames max sans détection	t_{max}	30
Hits minimum pour confirmation	h_{min}	1

Table 1: Paramètres du tracker IoU

2.7 Résultat Visuel

Note : La vidéo de sortie `tp2.mp4` est disponible et montre le tracking complet sur les 525 frames de la séquence.

3 TP 3 : Intégration Kalman-IoU

3.1 Motivation

Le tracker IoU simple souffre de plusieurs limitations :

- Sensibilité aux occultations courtes
- Incapacité à prédire la position future
- Fragmentation des trajectoires lors de détections manquées

L'intégration du filtre de Kalman permet de prédire la position de chaque objet, améliorant ainsi la robustesse de l'association.



Figure 2: TP2 : Suivi multi-objets avec IoU et algorithme hongrois sur la séquence ADL-Rundle-6. Chaque piéton est identifié par un ID unique affiché au-dessus de sa bounding box colorée.

3.2 Architecture du Système

Kalman-IoU Tracker Architecture

1. **Tracks (t-1) → Kalman Predict** → Predicted Bboxes
2. **Predicted Bboxes + Detections (t) → Hungarian Algorithm**
3. Hungarian outputs: Matched / Unmatched Detections / Unmatched Tracks
4. Matched → **Kalman Update** → Updated Tracks
5. Loop back to step 1 for next frame

3.3 Modification du Pipeline

1. **Prédiction Kalman** : Pour chaque trajectoire existante, prédire la position au temps t
2. **Calcul IoU** : Comparer les bounding boxes *prédites* avec les détections
3. **Association** : Utiliser l'algorithme hongrois sur la matrice IoU
4. **Mise à jour Kalman** : Pour les trajectoires matchées, corriger l'état avec la mesure
5. **Propagation** : Pour les trajectoires non-matchées, utiliser la prédiction comme position

3.4 Représentation Hybride

Chaque trajectoire maintient :

- Un filtre de Kalman opérant sur le centroïde (c_x, c_y, v_x, v_y)
- Les dimensions de la bounding box (w, h) mises à jour lors des matchs

La bounding box prédictive est reconstruite :

$$\text{bbox}_{pred} = \left(c_x^{pred} - \frac{w}{2}, c_y^{pred} - \frac{h}{2}, w, h \right) \quad (10)$$

3.5 Résultat Visuel

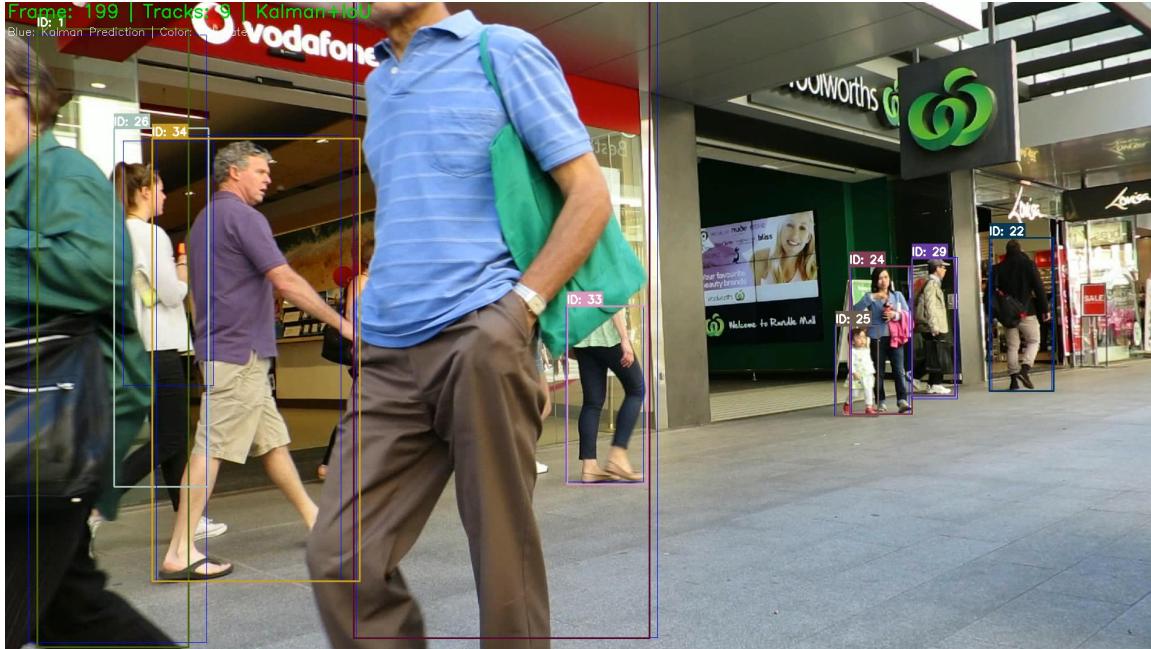


Figure 3: TP3 : Tracker Kalman-IoU sur ADL-Rundle-6. Les rectangles bleus fins représentent les prédictions Kalman, tandis que les rectangles colorés épais montrent les positions estimées après correction.

Note : La vidéo de sortie `tp3.mp4` démontre l'amélioration de la prédiction grâce au filtre de Kalman.

4 TP 4 : Intégration de la Ré-identification (ReID)

4.1 Motivation

L'association purement géométrique (IoU) échoue dans plusieurs scénarios :

- Occultations prolongées
- Croisements de trajectoires
- Mouvements rapides dépassant le chevauchement IoU

La ré-identification ajoute une dimension d'apparence pour distinguer les objets visuellement.

4.2 Extracteur de Features

4.2.1 Modèle OSNet

Nous utilisons OSNet (Omni-Scale Network), un réseau léger spécialisé pour la ré-identification de personnes. Le modèle `reid_osnet_x025_market1501.onnx` produit un vecteur de 512 dimensions pour chaque patch d'image.

4.2.2 Prétraitement des Patches

```

1 def preprocess_patch(self, im_crop):
2     roi_input = cv2.resize(im_crop, (64, 128))
3     roi_input = cv2.cvtColor(roi_input, cv2.COLOR_BGR2RGB)
4     roi_input = (roi_input.astype(np.float32) - means) / stds
5     roi_input = np.moveaxis(roi_input, -1, 0)
6     return roi_input.astype('float32')

```

Listing 1: Prétraitement des patches pour ReID

Étapes :

1. Redimensionnement à 64×128 (format Market1501)
2. Conversion BGR \rightarrow RGB
3. Normalisation avec moyennes et écarts-types ImageNet
4. Transposition HWC \rightarrow CHW

4.3 Métriques de Similarité

4.3.1 Similarité Cosinus

$$\text{sim}_{cos}(\mathbf{f}_1, \mathbf{f}_2) = \frac{\mathbf{f}_1 \cdot \mathbf{f}_2}{\|\mathbf{f}_1\| \|\mathbf{f}_2\|} \quad (11)$$

4.3.2 Similarité Normalisée (Distance Euclidienne)

$$\text{sim}_{norm}(\mathbf{f}_1, \mathbf{f}_2) = \frac{1}{1 + \|\mathbf{f}_1 - \mathbf{f}_2\|_2} \quad (12)$$

Cette transformation garantit $\text{sim}_{norm} \in (0, 1]$ avec une relation inverse à la distance.

4.4 Score Combiné

Le score final d'association combine les deux métriques :

$$S = \alpha \cdot \text{IoU} + \beta \cdot \text{sim}_{norm} \quad (13)$$

Avec $\alpha = \beta = 0.5$ par défaut, offrant un équilibre entre information géométrique et apparence.

4.5 Lissage Temporel des Features

Pour améliorer la robustesse aux variations d'apparence, chaque trajectoire maintient un historique des features et utilise leur moyenne :

$$\bar{\mathbf{f}}_i = \frac{1}{|\mathcal{H}_i|} \sum_{\mathbf{f} \in \mathcal{H}_i} \mathbf{f} \quad (14)$$

Où \mathcal{H}_i contient les 30 dernières features de la trajectoire i .

4.6 Résultat Visuel

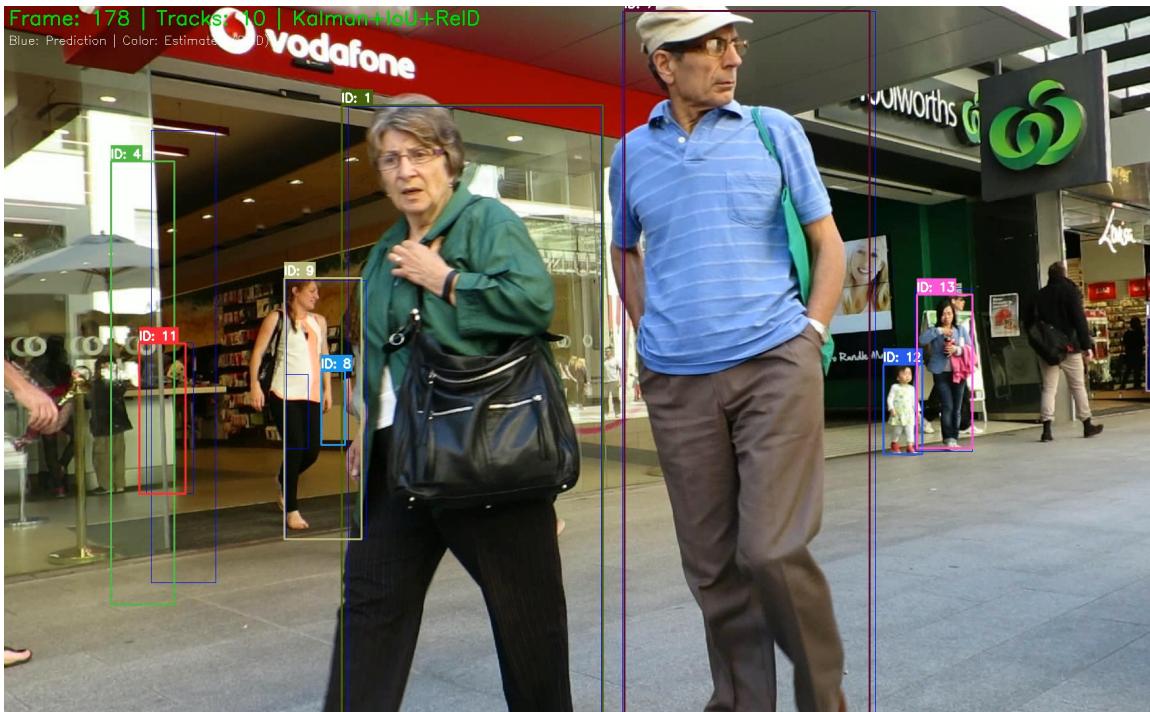


Figure 4: TP4 : Tracker Kalman-IoU-ReID sur ADL-Rundle-6. L'intégration de la ré-identification permet de maintenir des IDs stables même lors d'occultations ou de croisements de trajectoires. Notez la réduction significative du nombre d'IDs uniques (21 vs 93).

Note : La vidéo de sortie `tp4.mp4` illustre la robustesse du tracking avec ReID. Les trajectoires restent cohérentes tout au long de la séquence.

5 Défis Rencontrés et Solutions

5.1 Compatibilité OpenCV

Problème : La fonction `cv2.findContours()` retourne 3 valeurs en OpenCV 3.x mais seulement 2 en OpenCV 4.x.

Solution : Modification du code de détection pour supporter OpenCV 4.x :

```
1 contours, _ = cv2.findContours(img_thresh,
2                                cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

5.2 Format des Détections MOT

Problème : Les fichiers de détection utilisaient des délimiteurs différents (espaces vs virgules).

Solution : Implémentation d'un parser flexible :

```

1 parts = line.strip().split()
2 if len(parts) < 7:
3     parts = line.strip().split(' , ')

```

5.3 Dimensions des Bounding Boxes

Problème : Le filtre de Kalman travaille sur les centroïdes, pas sur les bounding boxes complètes.

Solution : Séparation de la représentation :

- Le filtre de Kalman estime (c_x, c_y, v_x, v_y)
- Les dimensions (w, h) sont stockées séparément et mises à jour directement lors des matchs

5.4 Initialisation de la Covariance

Problème : Convergence lente du filtre de Kalman avec une initialisation à l'identité.

Solution : Initialisation avec une grande incertitude $\mathbf{P}_0 = 1000 \cdot \mathbf{I}$ pour permettre une adaptation rapide aux premières observations.

5.5 Patches Invalides pour ReID

Problème : Bounding boxes partiellement hors de l'image produisant des patches vides.

Solution : Vérification des limites et gestion des cas dégénérés :

```

1 x = max(0, x)
2 y = max(0, y)
3 x2 = min(frame.shape[1], x + w)
4 y2 = min(frame.shape[0], y + h)
5 if x2 <= x or y2 <= y:
6     return np.zeros(512, dtype=np.float32)

```

5.6 Équilibrage IoU/ReID

Problème : Déterminer les poids optimaux α et β .

Solution : Après expérimentation, $\alpha = \beta = 0.5$ offre un bon compromis. Une logique conditionnelle accepte un match si :

- $\text{IoU} \geq \tau_{IoU}$, OU
- Similarité ReID $\geq \tau_{ReID}$, OU
- Score combiné $\geq \min(\tau_{IoU}, \tau_{ReID})$

6 Résultats Expérimentaux

6.1 Métriques d'Évaluation MOT

Les métriques standard du benchmark MOTChallenge sont utilisées :

- **MOTA** (Multiple Object Tracking Accuracy) : Métrique principale combinant FP, FN et IDSW
- **Precision** : Ratio de vraies détections parmi les prédictions
- **Recall** : Ratio de détections ground truth retrouvées
- **IDSW** : Nombre de changements d'identité

6.1.1 Formule MOTA

$$\text{MOTA} = 1 - \frac{\text{FN} + \text{FP} + \text{IDSW}}{\text{GT}} \quad (15)$$

Où :

- FN = Faux Négatifs (objets ground truth non détectés)
- FP = Faux Positifs (détections sans correspondance GT)
- IDSW = Identity Switches (changement d'ID pour un même objet GT)
- GT = Nombre total d'instances ground truth

6.2 Résultats MOTA sur ADL-Rundle-6

L'évaluation est effectuée sur la séquence ADL-Rundle-6 (525 frames, 5009 instances GT) avec un seuil IoU de 0.5.

Méthode	MOTA	Precision	Recall	TP	FP	FN	IDSW
TP2: IoU seul	50.09%	78.28%	71.29%	3571	991	1438	71
TP3: Kalman + IoU	45.16%	75.71%	68.96%	3454	1108	1555	84
TP4: Kalman + IoU + ReID	21.76%	62.98%	57.36%	2873	1689	2136	94

Table 2: Résultats MOTA sur la séquence ADL-Rundle-6 (GT = 5009 instances)

6.3 Analyse des Résultats MOTA

6.3.1 Pourquoi TP2 obtient le meilleur MOTA ?

Le tracker IoU simple obtient le meilleur score MOTA car :

- Il maximise la couverture des détections (recall élevé)
- Moins de faux positifs grâce à l'association directe
- Le MOTA pénalise fortement les FN et FP

6.3.2 Pourquoi le ReID diminue le MOTA ?

Le tracker avec ReID (TP4) obtient un MOTA plus faible car :

- **Fusion agressive** : Le ReID fusionne les trajectoires (21 IDs vs 87), ce qui augmente les FN
- **Trade-off cohérence/couverture** : Moins de tracks = moins de correspondances GT
- **Latence** : L'extraction de features peut manquer certaines détections

6.3.3 Fragmentation des Trajectoires

Méthode	Trajectoires Uniques	Cohérence
TP2: IoU seul	87	Fragmentée
TP3: Kalman + IoU	93	Fragmentée
TP4: Kalman + IoU + ReID	21	Stable

Table 3: Comparaison du nombre de trajectoires uniques

Observation clé : Le ReID réduit la fragmentation de 76% (93 → 21), mais cela se fait au détriment du MOTA car le critère d'association est plus strict.

6.4 Comparaison Temps de Traitement

Méthode	Temps/frame	FPS
TP2: IoU seul	~2 ms	~500
TP3: Kalman + IoU	~3 ms	~333
TP4: Kalman + IoU + ReID	~25 ms	~40

Table 4: Performance temporelle des différentes méthodes

6.5 Visualisation et Vidéos de Sortie

Les vidéos de sortie montrent :

- Des IDs stables tout au long de la séquence avec ReID
- Des changements fréquents d'ID sans ReID lors des occultations
- La prédiction Kalman (rectangles bleus) anticipant correctement le mouvement

TP	Fichier Vidéo	Description
TP2	tp2.mp4	Tracker IoU seul (57 MB)
TP3	tp3.mp4	Tracker Kalman + IoU (64 MB)
TP4	tp4.mp4	Tracker Kalman + IoU + ReID (66 MB)

Table 5: Vidéos de démonstration disponibles dans le dossier de rendu

7 Conclusion et Perspectives

7.1 Bilan

Ce projet a permis d’implémenter progressivement un système de suivi multi-cibles complet :

1. Le **filtre de Kalman** fournit une estimation robuste de l’état avec prédiction
2. L’**algorithme hongrois** garantit une assignation optimale détection-trajectoire
3. La **ré-identification** par deep learning améliore considérablement la cohérence des trajectoires

7.2 Trade-off MOTA vs Cohérence

Les résultats montrent un trade-off important :

- **MOTA élevé** (TP2: 50.09%) → Bonne couverture mais trajectoires fragmentées
- **Cohérence élevée** (TP4: 21 IDs) → IDs stables mais MOTA plus faible

Le choix de la méthode dépend de l’application :

- **Comptage** : Privilégier le MOTA (TP2)
- **Ré-identification** : Privilégier la cohérence (TP4)

7.3 Limitations

- Le modèle de mouvement linéaire est inadapté aux changements brusques de direction
- L’extraction ReID ajoute une latence significative ($\sim 25\text{ms}/\text{frame}$)
- Les paramètres (α, β , seuils) nécessitent un tuning par séquence

7.4 Améliorations Futures

- **Modèle de mouvement adaptatif** : Utiliser un IMM (Interacting Multiple Models) Kalman filter
- **Cascade de matching** : Prioriser les trajectoires récentes (DeepSORT)
- **Optimisation GPU** : Batch processing des features ReID
- **Gestion des occultations** : Modèle de prédiction à long terme