COMPUTER VISION

Suivi automatique d'un individu dans un environnement multi-caméras

Thomas GENTILHOMME

Télécom Paris - Option IA

Encadrant: Laurent CERVONI
Référent : Olivier FERCOQ

Table des matières

Introduction

- Domaine majeur de la Computer Vision et de l'IA.
- Nombreuses applications: surveillance, sécurité, marketing, militaire, ...
- Nombreux défis : qualité des détections, reconnaissance d'individus au fil des images, problèmes d'occlusions, scènes bondées, temps d'exécution...
- Enjeux éthiques : applications, données personnelles.
- *Tracking-by-detection*.

R-CNN : Region Based Convolutional Neural Networks

- Architecture en **deux** étages : une partie de **proposition de régions** et un **réseau de détection**.
- Propositions de région : $\approx$ 2000 : *algorithme de recherche sélective*.
- Détections : **extraction** des features avec un CNN, **classification** avec un SVM et **régression** pour les coordonnées.
- **Inconvénients :** très lent (47s/image avec VGG16), pas d'apprentissage au niveau de l'algorithme de recherche sélective.

R-CNN : Region Based Convolutional Neural Networks

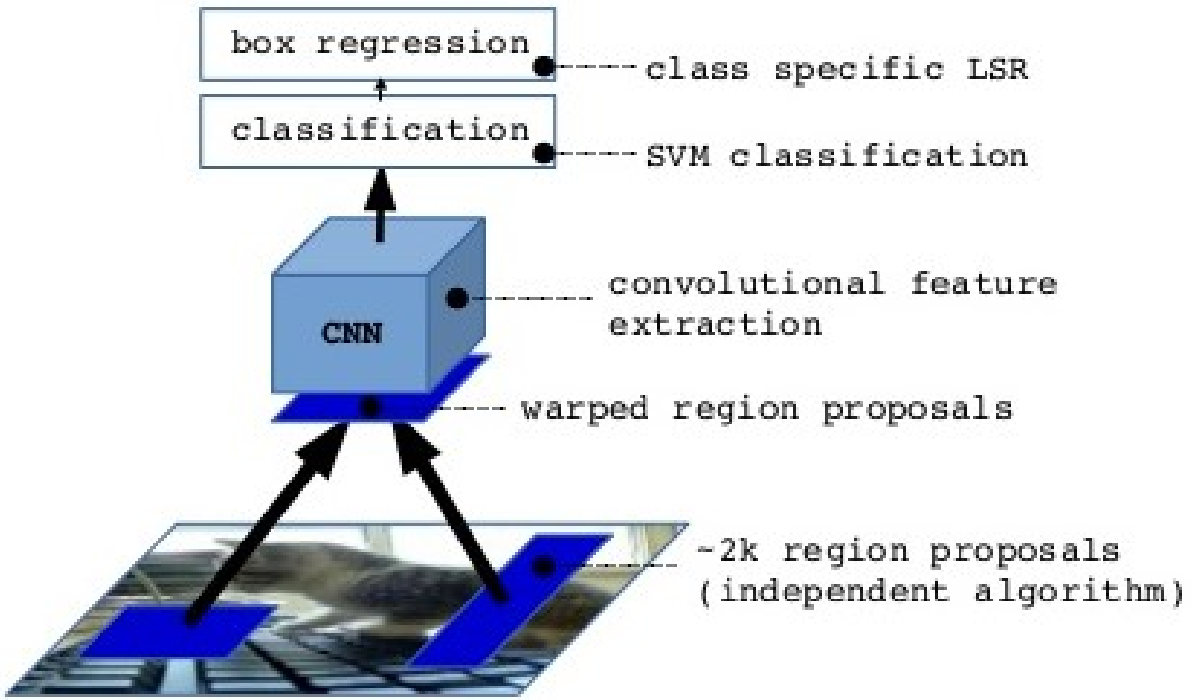


Figure 1: Architecture d'un R-CNN.

Fast R-CNN

- Architecture très similaire à celle de *R-CNN*.
- L'extraction des features s'effectue désormais en amont de la recherche de régions, qui s'effectue alors sur cette *feature map*.
- Ajout d'une couche de *RoI Pooling* afin d'obtenir des features de même dimensions. *(explication dans la partie suivante)*
- Le SVM est remplacé par une classification *Softmax*.
- **Avantages :** beaucoup plus rapide que l'architecture *R-CNN*. Meilleure classification avec le *softmax*.
- **Inconvénient :** temps de calcul dominé par l'étage de propositions de région ($\approx 86\%$). De plus, cet étage ne participe pas à l'apprentissage.

Fast R-CNN

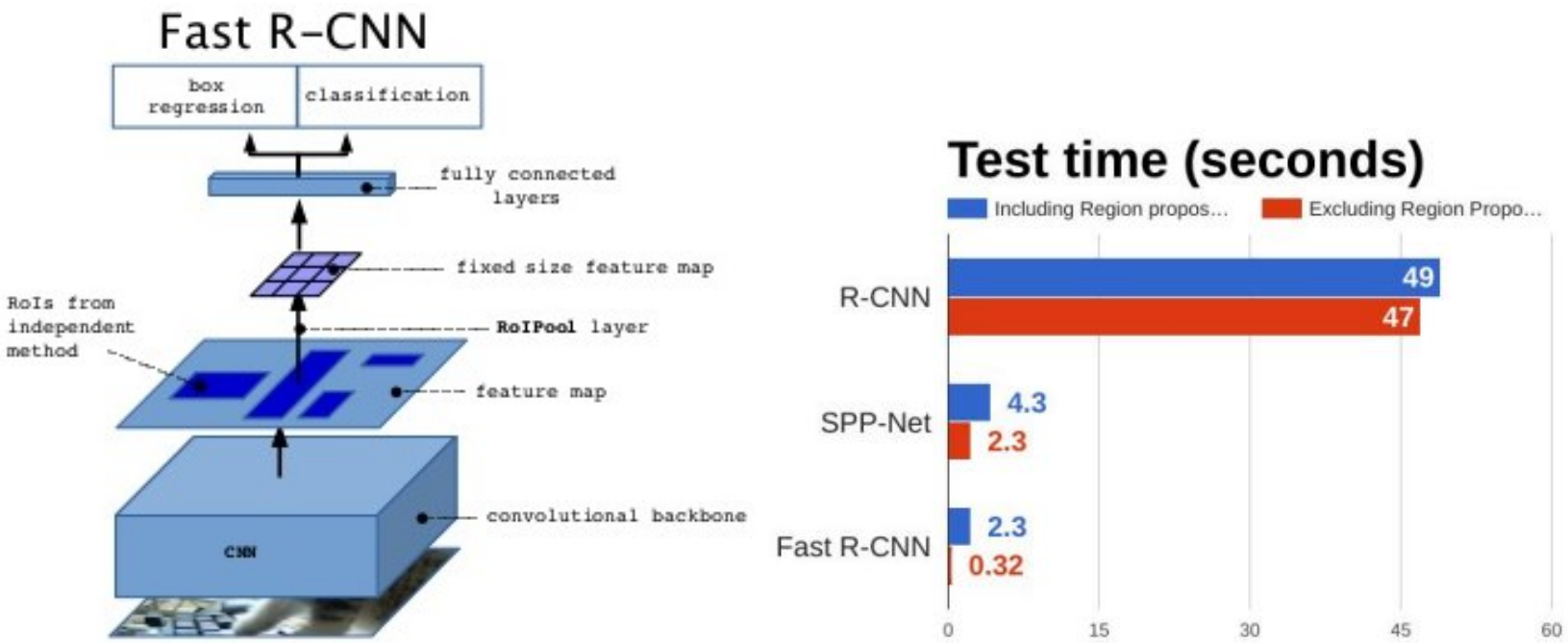


Figure 2: Architecture d'un Fast R-CNN. Le temps de détection est dominé par les propositions de région.

Faster R-CNN

- Les régions d'intérêt sont désormais proposées avec un **CNN**.
- Tâche effectuée par le **Region Proposal Network (RPN)** : son rôle est de prédire des régions à partir des features de l'image.
- L'étage de propositions de région participe maintenant à l'entraînement.
- Le réseau de détection **(DN)** demeure similaire à celui de *Fast R-CNN*.
- 4 losses prennent par à l'entraînement : 2 pour le **RPN**, 2 pour le **DN**.
- **Amélioration :** le temps de détection est passé à 0.2s/image (**5 FPS**).

Faster R-CNN : RPN (Region Proposal Network)

- Extraction d'une feature map avec la couche 5 d'un *ResNet*.
- Celle-ci est parcourue par une fenêtre glissante de taille $3x3$.
- A chaque localisation 9 propositions de régions (*anchors*) sont faites (3 échelles et 3 ratios différents).
- Chaque proposition passe au travers d'une **couche de classification** et d'une **couche de régression** :

$$L_{RPN}(p_i, t_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*)$$

- L_{cls}: cross-entropy-binaire. L_{reg}: Smooth-L1.
- p_i = probabilité que l'*anchor i* contienne un objet.
- $p_i^* = 1$ signifie que l'*anchor i* est labelisé **positif**.

Faster R-CNN : RPN (Region Proposal Network)

- Un *anchor* est **labelisé positif** s'il chevauche suffisamment n'importe quelle vrai objet.
- t_i est le vecteur de coordonnées de l'*anchor i*.
- Enfin, l'algorithme **NMS** (Non-Maximum Suppression) est appliqué pour réduire le nombre de régions proposées d'$\approx$ 6000 à $\approx$ 300 en regroupant les régions voisines.

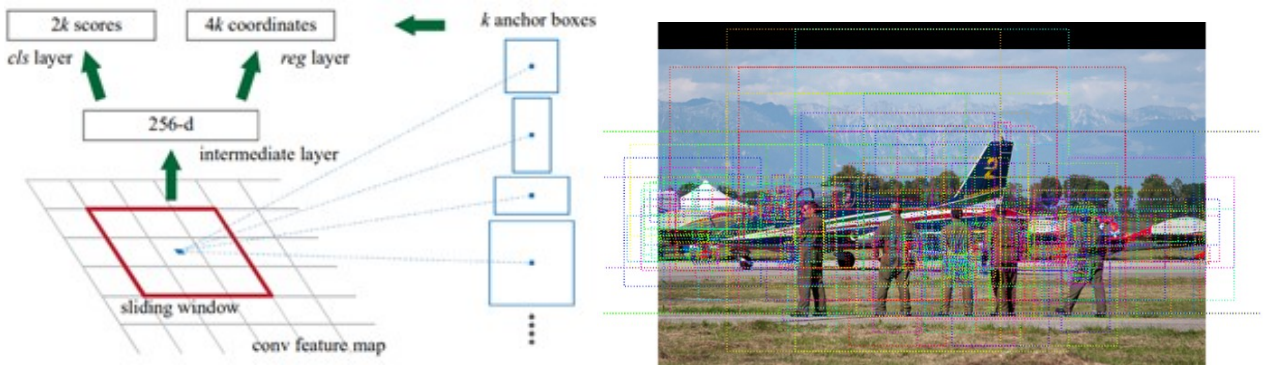
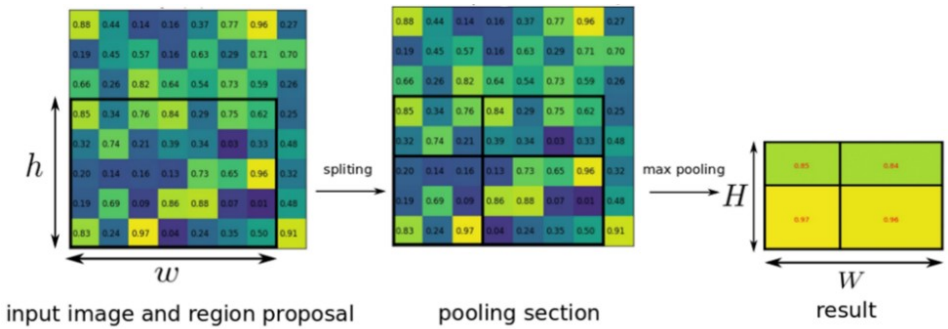


Figure 3: Architecture du RPN. Exemple des nombreux *anchors* proposés.

Faster R-CNN : DN (Detection Network)

Cette couche se caractérise par la présente d'une *RoI Pooling Layer*. Chaque région proposée est de taille différente, cette couche permet de toutes les redimensionner dans une taille unique *HxW*.



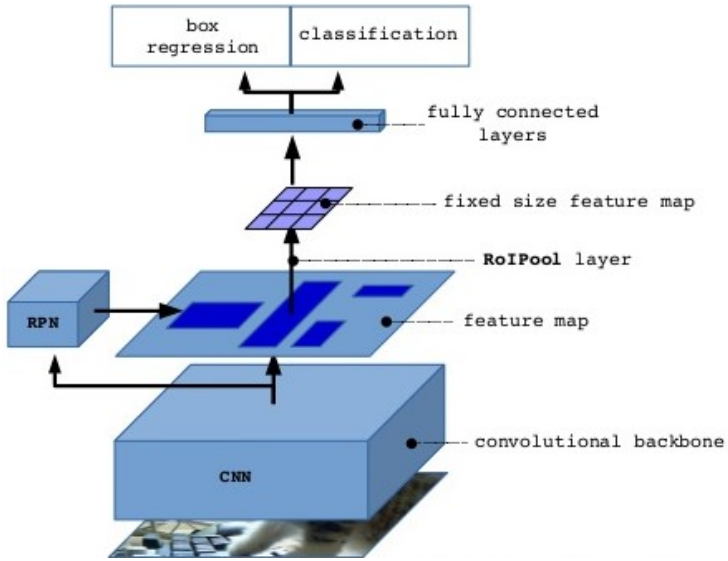
- Loss multi-tâches :

$$L_{DN}(p, u, t^u, v) = L_{cls}(p, u) + \lambda[u \geq 1]L_{loc}(t^u, v)$$

- p, u classe prédite/réelle ; t^u, v coordonnées prédites/réelles.
- $[u \geq 1] = 1 \iff u \geq 1$ ($u = 0$ étant la classe *background*).

Faster R-CNN

Faster R-CNN est entraîné de façon à ce que les deux sous-réseaux partagent des couches de convolution et puissent former un réseau unifié.



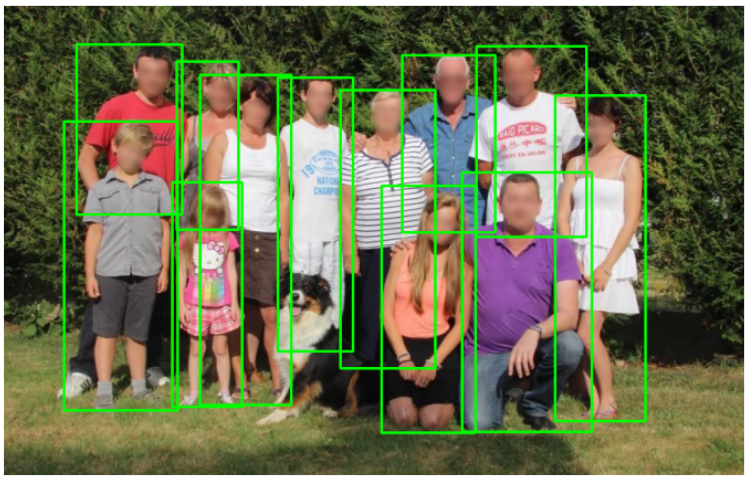
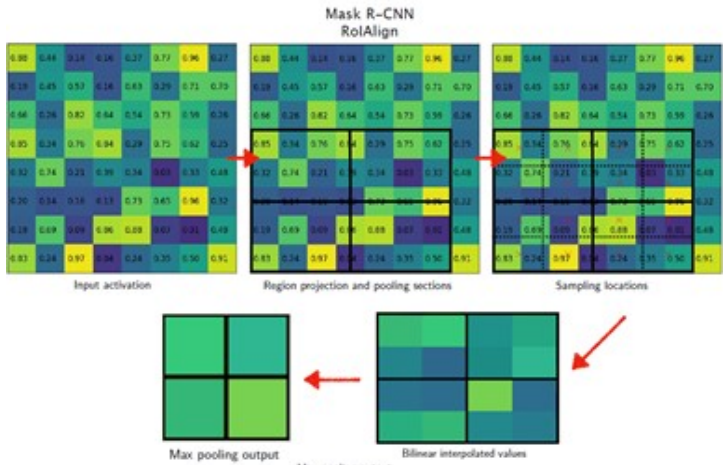


Figure 4: Architecture d'un Faster R-CNN. Exemple de détection de personnes sur une photo personnelle.

Mask R-CNN

- Permet de faire de la **segmentation d'instance** (*i.e.* générer des masques binaires).
- Ajout d'une autre branche en sortie de la couche pooling.
- Ajout d'une *loss* L_{mask} (K *sigmoïdes*/pixel, K = classes).
- Remplacement de la couche *RoIPooling* par *RoIAlign*. Cette couche évite d'arrondir le nombre de pixels lors du redimensionnement et donc de désaligner le masque.



Mask R-CNN

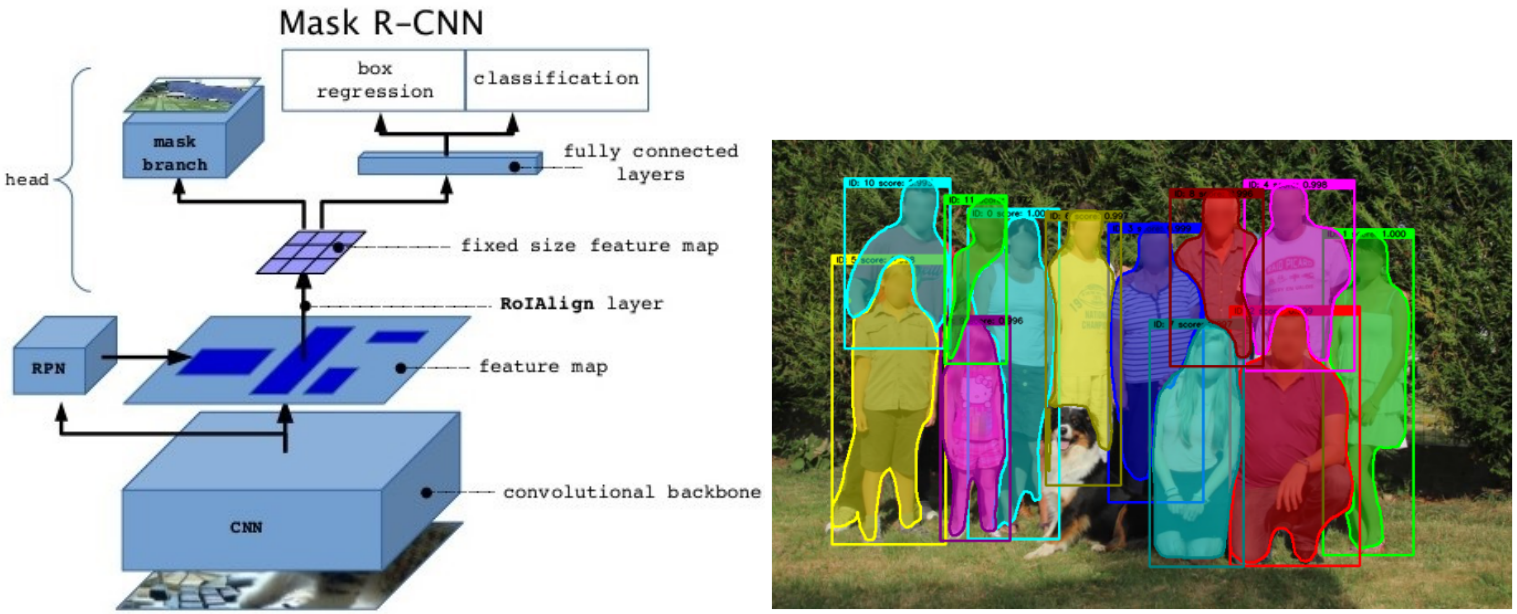


Figure 5: Architecture d'un Mask R-CNN. Exemple de détection+segmentation de personnes sur une photo personnelle.

DeepSORT

- **Principe :** associer à chaque *frame* de la vidéo les individus (*tracks*) en mémoire avec les nouvelles détections.
- Méthode **online**, rapide et performante.
- **Track :** représentation d'un individu sur une vidéo entière. Il a un **ID** unique.
- **Boite englobante prédite :** boite englobante d'un track à l'instant présent prédite par un **filtre de Kalman**.
- **Boite englobante détectée :** boite englobante d'un individu à l'instant présent générée par le détecteur *Mask R-CNN*.
- **Descripteur d'apparence :** permet de représenter chaque détection par un **vecteur riche** en extrayant leur features avec un CNN pré-entrainé sur une tache de Re-Identification avec la **distance cosinus**.

DeepSORT - Suppression et création de tracks

Quand un individu entre ou sort de l'image, son track associé doit être soit créé soit supprimé.

- Le track d'un individu est **supprimé** après A_{max} non-association.
- A chaque détection non-associé à un track existant, un nouveau track **provisoire** est créé. Après n_{init} associations consécutives réussies, il devient **confirmé**. Sinon, il est **supprimé**.

DeepSORT - Résoudre le problème d'association

- Problème résolue avec l'**algorithme Hongrois**.
- Mesure de coût entre le i-ième track et la j-ième détection :
  - Distance de Mahalanobis : $d^{(1)}(i,j) = (d_j - y_i)^T S_i^{-1}(d_j - y_i)$
  - Distance cosinus : $d^{(2)}(i,j) = min\{1 - r_j^T r_k^{(i)} | r_k^{(i)} \in R_i\},\ R_i = \{r_k^{(i)}\}_{k=1}^{100}$
- Variables binaires pour éliminer les associations supérieures à un seuil:

$$b_{i,j}^{(1)} = 1[d^{(1)}(i,j) \leq t^{(1)}]; b_{i,j}^{(2)} = 1[d^{(2)}(i,j) \leq t^{(2)}]$$

- Critères d'association : combination de ces deux mesures :

$$c_{i,j} = \lambda d^{(1)}(i,j) + (1-\lambda)d^{(2)}(i,j); b_{(i,j)} = b_{i,j}^{(1)}.b_{i,j}^{(2)}$$

DeepSORT - Résoudre le problème d'association

L'algorithme de **Matching Cascade** réalise les associations entres les tracks et les détections, en privilégiant les tracks les plus récemment mis à jour, et retourne une liste d'associations réussies $\mathcal{M}$ et une liste de détections non-réussies $\mathcal{U}$.

Listing 1 Matching Cascade

Input: Track indices $\mathcal{T} = \{1, \ldots, N\}$, Detection indices $\mathcal{D} = \{1, \ldots, M\}$, Maximum age $A_{\max}$

Compute cost matrix $\boldsymbol{C} = [c_{i,j}]$ using Eq. 5
Compute gate matrix $\boldsymbol{B} = [b_{i,j}]$ using Eq. 6
Initialize set of matches $\mathcal{M} \leftarrow \emptyset$
Initialize set of unmatched detections $\mathcal{U} \leftarrow \mathcal{D}$
for $n \in \{1, \ldots, A_{\max}\}$ **do**
  Select tracks by age $\mathcal{T}_n \leftarrow \{i \in \mathcal{T} \mid a_i = n\}$
  $[x_{i,j}] \leftarrow \text{min\_cost\_matching}(\boldsymbol{C}, \mathcal{T}_n, \mathcal{U})$
  $\mathcal{M} \leftarrow \mathcal{M} \cup \{(i,j) \mid b_{i,j} \cdot x_{i,j} > 0\}$
  $\mathcal{U} \leftarrow \mathcal{U} \setminus \{j \mid \sum_i b_{i,j} \cdot x_{i,j} > 0\}$
end for
return $\mathcal{M}, \mathcal{U}$

Un dernier algorithme d'association est appliqué sur les détections de $\mathcal{U}$ d'âge $n = 1$: l'**Intersection over Union**: $IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union}$.

Généralités

- Architecture *end-to-end* de *tracking-by-detection*.
- L'algorithme peut utiliser n'importe quelle vidéo, n'importe quel détecteur et n'importe quel traqueur à partir où le format de sortie est adapté (*Ex: format des coordonnées des boites, classe "individu" du set d'entraînement du détecteur, ...*).
- Implémentation Mask R-CNN:
  - Open-source et pré-entraîné sur le dataset *MS COCO*.
  - Quatre listes : *rois* ; *class_ids* ; *scores* ; *masks*.
- Implémentation DeepSORT:
  - Open-source et encoder pré-entraîné sur le dataset de Re-ID *MARS*.
  - Code adapté à notre architecture et à l'association multi-caméras.

Classes implémentées

`https://github.com/Thomas-Gentilhomme/AI-Project/blob/main/Multi_targets_multi_cameras_tracking_MaskRCNN_DeepSORT.ipynb`

- **MaskRCNN**: créé une instance du modèle. Réaliser les détections.
- **DeepSORT**: créé une instance du traqueur. Réaliser les associations.
- **Individuals**: représente les caractéristiques de tous les individus d'une frame.
- **Track**: réalise le track complet d'une vidéo et renvoie la vidéo dessinée.
- **MultiTrack**: réalise le track complet d'un ensemble de vidéos et renvoie les vidéos dessinées.

Fonctions implémentées

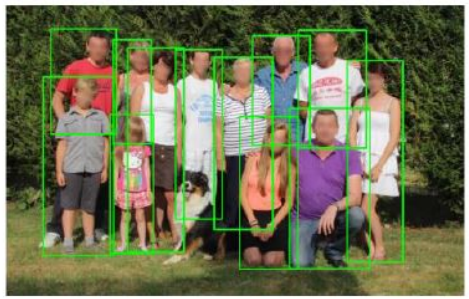
`https://github.com/Thomas-Gentilhomme/AI-Project/blob/main/functions.py`

- **update_results**: ne garder que les individus dont la confiance est au-delà d'un seuil.
- **convert_roi_shape**: changer le format des coordonnées des boites.
- **draw_without_tracking** et **draw_with_tracking**: réaliser les dessins.
- **candidates** renvoie les IDs d'individus similaires à un ID *query*.
- **crop_box** renvoie l'image contenue dans boite englobante.
- **display_crops** affiche les individus similaires à un individu *query*.

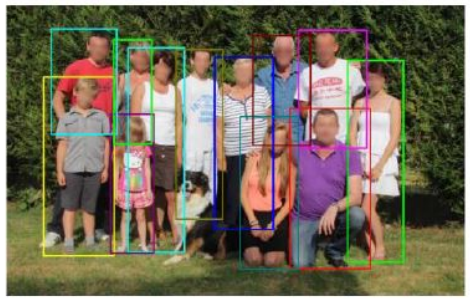
Photo

- Une **photo** personnelle de **12 individus** différents et un non-individu.

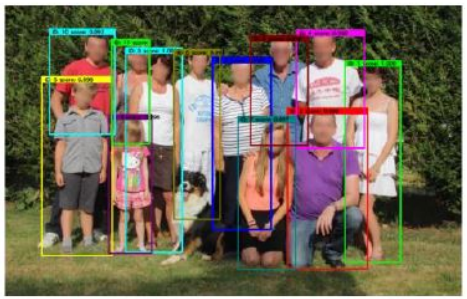
Image d'origine



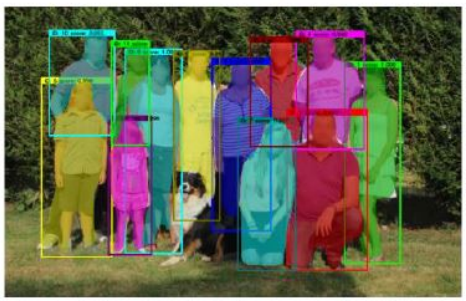
Avec boites englobantes



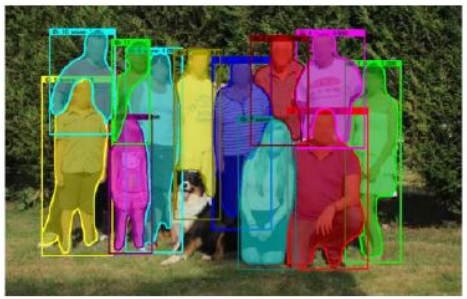
Avec boites en couleurs différentes



Avec captions



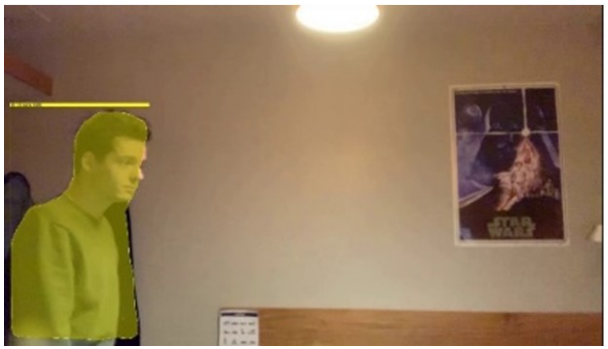
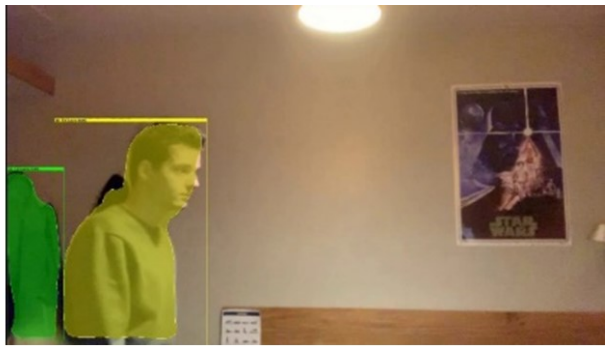
Avec masques



Avec contours de masques

Vidéos

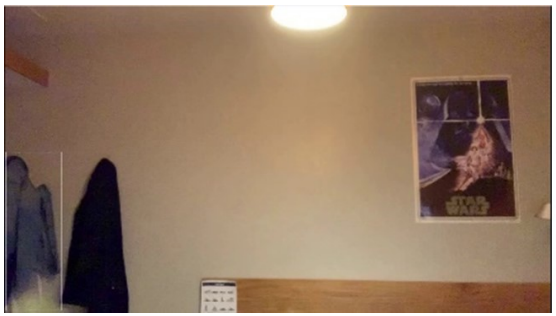
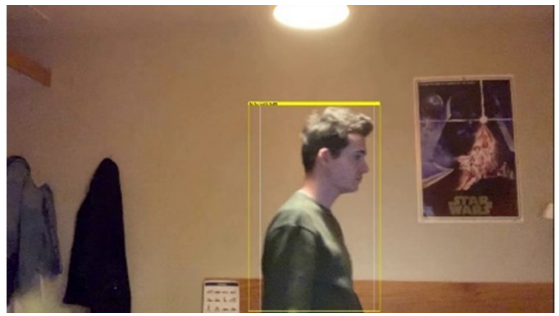
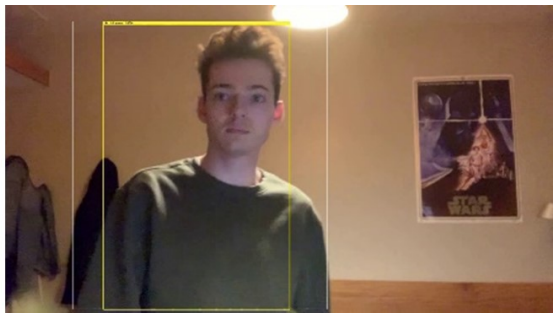
- Une **vidéo** personnelle de **14s** contenant **1** individu et deux courtes occlusions.

- Une **vidéo** personnelle de **20s** contenant une **dizaine** d'individus, très petits, et plusieurs occlusions.


Analyse de la qualité de la détection et du tracking

- Sur la vidéo avec **un unique individu**:
  - **Observations:** très peu d'erreurs de détection. Causes : objets inanimés de forme humaine. Tracking perturbé par les occlusions.
  - **Correction:** augmenter le **seuil de détection** à 0.98, A_{max} à 60 et n_{init} à 10.
  - **Résultat:** parfait. L'individu garde l'identifiant 1 du début à la fin. Les rares mauvaises détections ne sont plus traquées (boite blanche sur l'image 1).

Analyse de la qualité de la détection et du tracking

- Sur la vidéo avec **plusieurs petits individus**:
  - **Observations:** plusieurs erreurs de détection. Causes : objets très petits et sombres. Tracking perturbé par les occlusions. Le système est **instable**.
  - **Correction:** diminuer le **seuil de détection** à 0.6, augmenter A_{max} à 60 et n_{init} à 10.
  - **Résultat:** le système gagne en **robustesse** mais le résultat est loin d'être parfait. Les individus fixes et non-occultés sont bien traqués.

Analyse du temps d'exécution

Observations:

- Tracking très performant sur ce critère.
- Fréquence correcte mais plus faible que celle de l'état de l'art ($\approx$ 5 FPS).
- Fréquence inversement proportionnelle au nombre d'individus à traquer.
- Le temps de calcul est massivement consommé par le tracking et la fonction de dessin.

	Vidéo test 1	Vidéo test 2
Détection	80.01%	44.36%
Tracking	1.52%	2.46%
Dessin	12.10%	49.36%
Fréquence	**1.86 FPS**	**0.96 FPS**

Table 1: Sur Google Colaboratory.

Analyse du temps d'exécution

- **Observations:** le temps de dessin est dominé par le dessin des masques.
- **Correction:** retrait de l'affichage des masques et de leurs contours qui n'ont qu'un intérêt esthétique.
- **Résultat:** amélioration du temps de calcul, moins dépendant du nombre d'individus et désormais concentré $\approx 85\%$ dans la détection.

	Photo test 12 individus
Temps copie	1.00%
Temps boites	0.09%
Temps captions	0.09%
Temps masques	**47.02%**
Temps contours	**51.52%**

Table 2: Analyse du temps d'exécution de la fonction *draw_with_tracking()*.

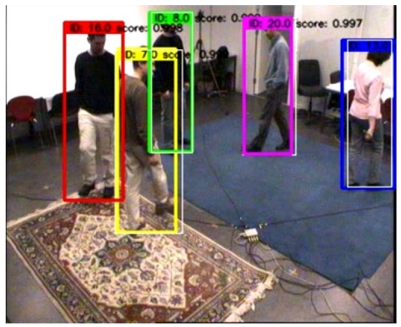
Analyse du temps d'exécution

	Avec masques		Sans masques	
	Vidéo test 1	Vidéo test 2	Vidéo test 1	Vidéo test 2
Détection	80.01%	44.36%	**87.96%**	**83.83%**
Tracking	1.52%	2.46%	1.77%	4.67%
Dessin	12.10%	49.36%	1.38%	0.68%
Fréquence	**1.86 FPS**	**0.96 FPS**	**2.01 FPS**	**1.78 FPS**

Table 3: Sur Google Colaboratory.

Présentation du dataset

4 vidéos d'**1 minute 58**, à **25 FPS**, sur lesquelles sont présents **6 individus**. Résolution: *360x288*.

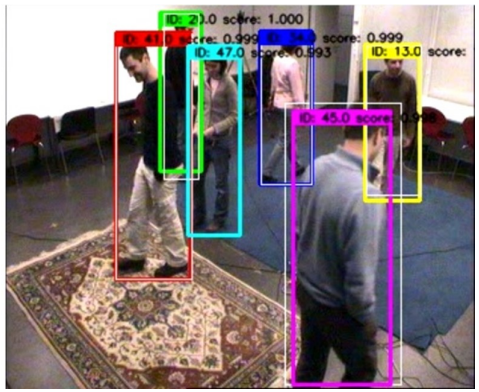

Temps d'exécution total: **65 minutes et 58 secondes**. En détail :

	Détection	Tracking	Dessin	Fréquence
Moyenne sur les 4 vidéos	95.44%	4.01%	0.15%	**2.99 FPS**

Paramètres: $A_{max} = 50$ (**2** secondes), $n_{init} = 5$, $max_{cosine\_distance} = 0.2$, $max_{IoU} = 0.7$, seuil de détection $= 0.95$ (compromis).

Analyse et optimisation

- **Observations :** excellentes détections, même lorsqu'un individu est partiellement caché par un autre. Tracking globalement bon mais perturbé par des **changements d'identité** et des **occlusions**, d'où une multiplication des tracks.



- **Corrections :** augmenter $max_{cosine\_distance}$ à 0.4 (compromis entre switches et occlusions) ainsi que n_{init} à 10, A_{max} à 100 (**4s**) et max_{IoU} à 1 (même compromis entre tracking "strict" et "laxiste").
- **Résultats :** diminution du nombre de tracks et de switches.

Cas particulier et Re-Identification

- **Cas particulier:** 4 caméras filmant une **même scène**.
- **Re-Identification:** méthode consistant à produire un classement de similarité entre différentes instances d'individus en réponse à une *query* (l'individu recherché). Pour cela, on utilise un réseau CNN entraîné sur une tâche de Re-ID (un **réseau Siamois**) avec une métrique précise et une **loss triple**.
- **N-plus-proches-voisins**: méthode pour retrouver les individus les plus similaires.

Implémentation

- **DeepSORT:** réutilisation des features utilisées par DeepSORT.
  On possède déjà des représentations riches des individus entraînés sur une tâche de Re-ID avec la **distance cosinus**.
- **Implémentation:**
  - Modification de DeepSORT pour récupérer les features.
  - Création d'une classe **MultiTracking**.
  - Association de chaque couple (cam, ID) avec son **feature moyen**.
  - Calcul d'une **matrice des distances** basée sur la **distance cosinus**.
  - Fonction *candidates()* qui renvoie les N plus proches individus d'un individu d'une vidéo dans une autre (ou bien de la **même caméra pour résoudre les problèmes de switches d'ID**).

Implémentation

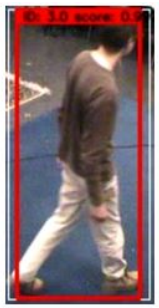
- **Résultat:**

```
        ##### QUERY #####

Camera:0 ; ID:3

        ##### CANDIDATES #####

ID candidates for cam 1: 52 15 30 50 31
ID candidates for cam 2: 43 4 30 46 11
ID candidates for cam 3: 9 3 1 6 15
```



QUERY : *caméra* 0 ; *ID* 3.

Caméra 1 ; *IDs* 52, 15, 20, 50, 21.

Caméra 3 ; *IDs* 9, 3, 1, 6, 15.

- **Problème:** l'algorithme ne renvoie que des identifiants. Comment prendre en compte les **problèmes de changement d'identité** ?

Problème des changements d'identité (switches)

- **Idée:** associer à chaque track **l'individu le plus représentatif**.
- **Implémentation:**
  - Modification de DeepSORT pour récupérer chaque *crop* d'individu (*i.e.* l'image au sein de son cadre) avec la fonction *crop_box()*.
  - Éliminations des images trop petites.
  - Re-dimensionner chaque image pour pouvoir calculer une **image moyenne**.
  - Associer à chaque ID l'image ressemblant le plus à son image moyenne.

Problème des changements d'identité (switches)

- **Résultat:**

QUERY : *caméra* 0 ; *ID* 3.

Caméra 1 ; *IDs* 52, 15, 30, 50, 31.

Caméra 2 ; *IDs* 43, 4, 30, 46, 11.

Caméra 3 ; *IDs* 9, 3, 1, 6, 15.

- **Observations:** correct sans pour autant être optimal. Mais très rapide.
- **Critiques:**
  - Critère plus représentatif pour calculer l'image moyenne.
  - Critère plus discriminant pour trouver l'image la plus représentative.

Conclusion et pistes d'amélioration

- Bien que **plus lent** que d'autres détecteurs (ex: *YOLO*), **Mask R-CNN** donne d'**excellents résultats**. Un seuil de détection très élevé permet de toujours correctement détecter des individus partiellement occultés ou très petits tout en évitant la fausse détection.
- **DeepSORT** est très efficace, d'autant plus que les détections en amont sont bonnes, et très rapide. Son principal défaut est le **traitement des occlusions**, la méthode n'est pas assez robuste. Plus il y aura d'individus, plus DeepSORT aura des difficultés à traquer efficacement .
- **Entraînement joint de Mask R-CNN et de l'encoder du Deep SORT pour partager des couches de convolution.**

Conclusion et pistes d'amélioration

- Utiliser les features de DeepSORT pour le **multi-tracking** est un gain de temps considérable mais d'autres extracteurs comme **AlignedReID** pourraient donner de meilleurs résultats.
- Finalement, les caractéristiques de la (des) scène(s) filmée(s) sont déterminantes. Elles sont à prendre en compte pour optimiser les paramètres mais également pour l'apprentissage.
- Domaine de recherche populaire et en constant progrès qui ne doit pas faire oublier les enjeux éthiques qui y sont liés.

Conclusion

Remerciements:
Laurent CERVONI et Olivier FERCOQ