YOLO: Unified, Real-Time Object Detection
&
Seq-NMS for Video object Detection

Yunyun SUN
Sixiang XU
Yutong YAN
Heng ZHANG

télécom saint-étienne
école d'ingénieurs nouvelles technologies

Sommaire

- Problématique
- Cahier des charges
- La réalisation
  - YOLO: Unified,Real-Time Object Detection
  - Seq-NMS for Video object Detection
  - Architecture du système
- Bilan
  - Analyse de résultat
  - Atouts et limites

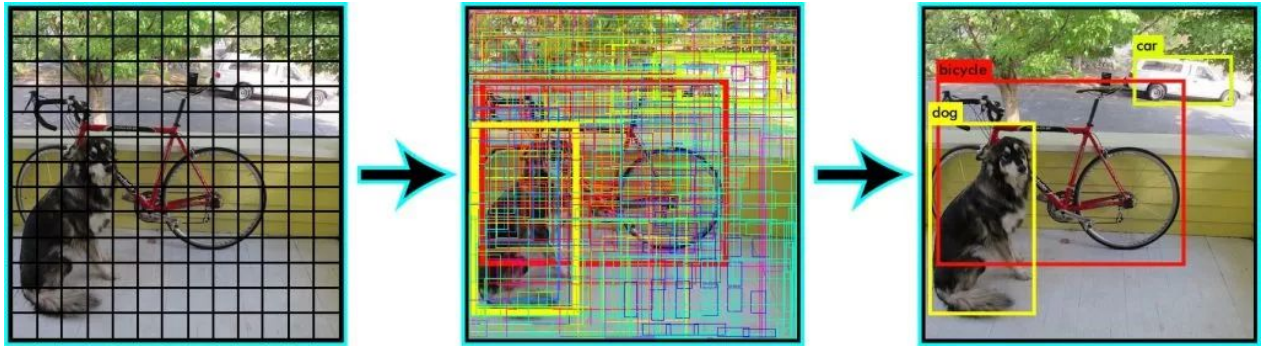
Problématique

Problématique

Les performances de l'état de l'art en détection d'objets sur une image ont été considérablement améliorées au cours des deux dernières années.

D'une part, **l'évolution des CNN**(AlexNet, InceptionNet, ResNet, etc) a amélioré la capacité d'extraction des features sur une image.

De l’autre part, le développement sur **l’approche de détection**(Faster R-CNN, R-FCN, SSD, YOLO, etc) permet de mieux utiliser ces features et augmenter la précision et la vitesse.



Problématique

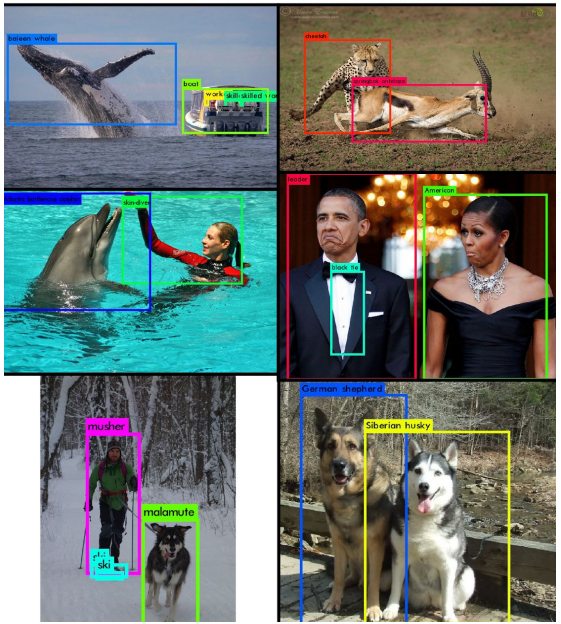
La différence entre la détection d'image et la détection de la vidéo est **les informations temporelles et contextuelles**.

Cahier des charges

Cahier des charges

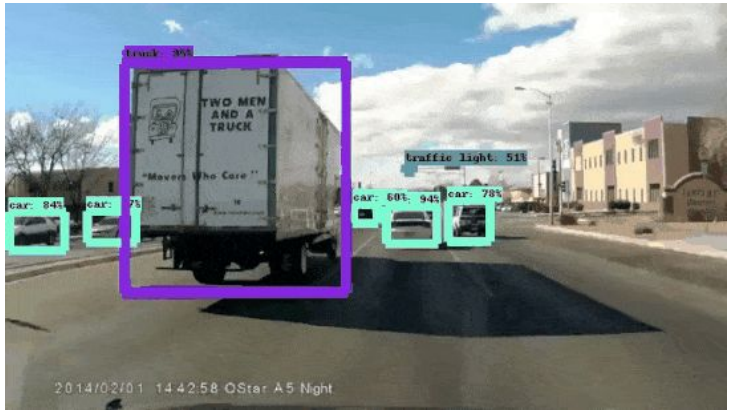
Notre projet vise à réaliser **un système de détection et de tracking d'objet dans la vidéo.**

Concrètement, nous avons conservé la framework de **détection d'image(YOLO)** autant que possible et ajouté un **post-traitement(Seq-nms)** pour améliorer la robustesse de mouvement d'objet.



Étapes principles

1. Extraction d'images à partir d'une vidéo;
2. Effectuer la détection sur chaque frame extrait en utilisant l'approche **YOLO**;
3. Filtrer les résultats de l'étape précédente avec l'algorithme **seq-nms**.
4. Reconstruire la vidéo à partir des résultats de détection filtrées.



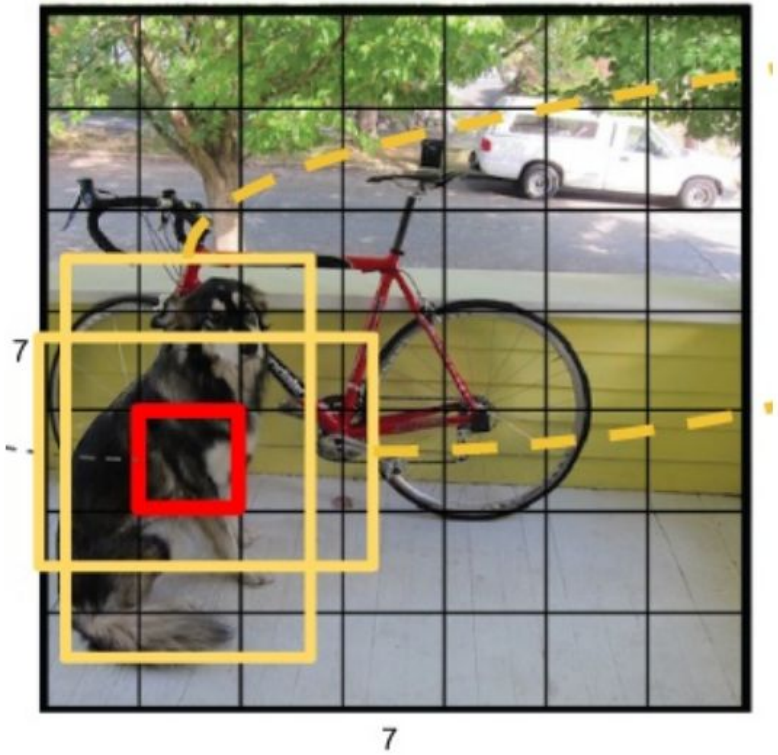
La réalisation

La réalisation:
YOLO: Unified,Real-Time Object Detection

- YOLO est un réseau de neurones intelligent pour **la détection d'objets en temps réel.**
- Pour la **vraie vidéo détection**: Seq-nms sur cette base
- **L'état de l'art** pour être un détecteur d'objet.
- **YOLO ne regarde qu'une seule fois l'image** (d'où son nom: You Only Look Once) mais d'une manière intelligente

YOLO

- YOLO divise l'image en une grille de **13** par **13** cellules
- Chacune des cellules est responsable de prédire **5** bounding boxes
- Un bounding box décrit le rectangle qui entoure un objet
- YOLO donne en sortie un **score de confiance** qui nous indique si les bounding box prédites incluent effectivement un objet

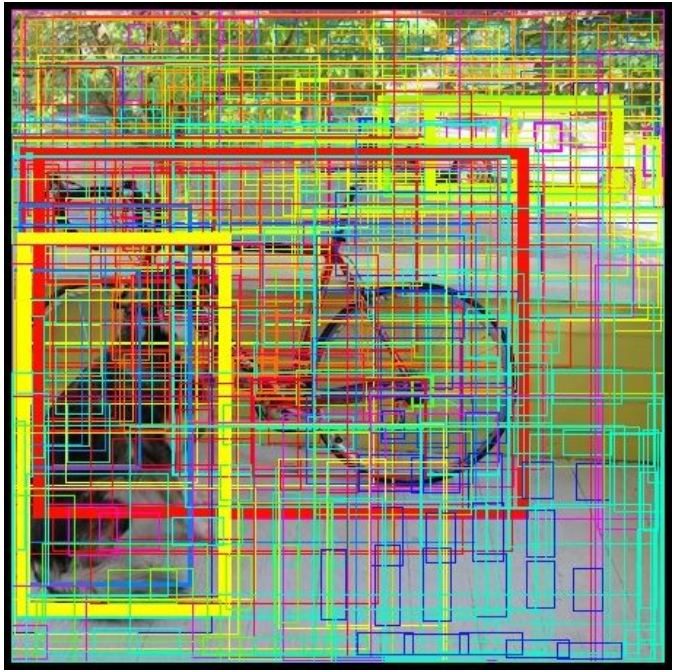


YOLO

- Le score ne dit rien sur la classe d'objet dans la boîte, juste **si la forme de la boîte est bonne**
- Les bounding box prédites peuvent ressembler à l’image ci-contre
- Pour chaque bounding box, **la cellule prédit également les classes**
- Base d’image: **COCO**, 80 classes

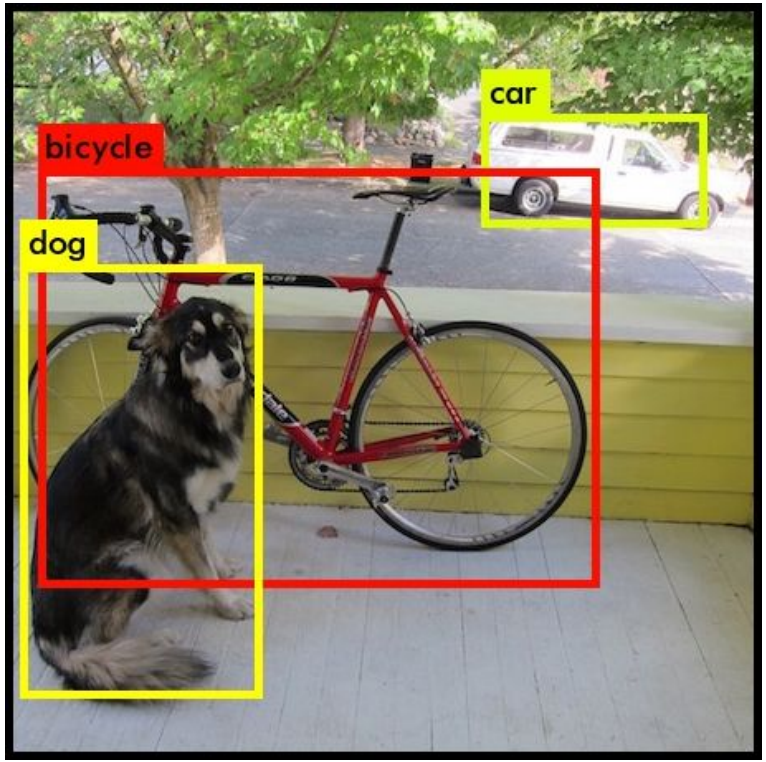
YOLO

- **Score de confiance** + **score de la prediction** => **score final** -> **quel type de l'objet**
- La grosse boîte jaune sur la gauche est à 85% sûre qu'elle contient l'objet "chien"
- 845 bounding box au total:
  - **13 × 13 = 169** cellules
  - chaque cellule prédit **5** bounding boxes
- La plupart de ces boîtes ont des scores de confiance très bas: seuil de 30%



YOLO

- La prediction finale
- 845 prédictions en meme temps
- Le réseau de neurones ne fonctionne qu'une seule fois (You Only Look Once)



YOLO

- L'architecture de YOLO est très simple
- 125 canaux pour chaque cellule:
  - Les données pour les bounding box
  - les prédictions de classe
  - 5 bounding boxs * 25 éléments de données
    - x et y, largeur, hauteur du rectangle de la boite
    - le score de confiance
    - la distribution de probabilité sur les classes

Layer	kernel	stride	output shape
Input			(416, 416, 3)
Convolution	3×3	1	(416, 416, 16)
MaxPooling	2×2	2	(208, 208, 16)
Convolution	3×3	1	(208, 208, 32)
MaxPooling	2×2	2	(104, 104, 32)
Convolution	3×3	1	(104, 104, 64)
MaxPooling	2×2	2	(52, 52, 64)
Convolution	3×3	1	(52, 52, 128)
MaxPooling	2×2	2	(26, 26, 128)
Convolution	3×3	1	(26, 26, 256)
MaxPooling	2×2	2	(13, 13, 256)
Convolution	3×3	1	(13, 13, 512)
MaxPooling	2×2	1	(13, 13, 512)
Convolution	3×3	1	(13, 13, 1024)
Convolution	3×3	1	(13, 13, 1024)
Convolution	1×1	1	(13, 13, 125)

La réalisation:
Seq-NMS for Video object Detection

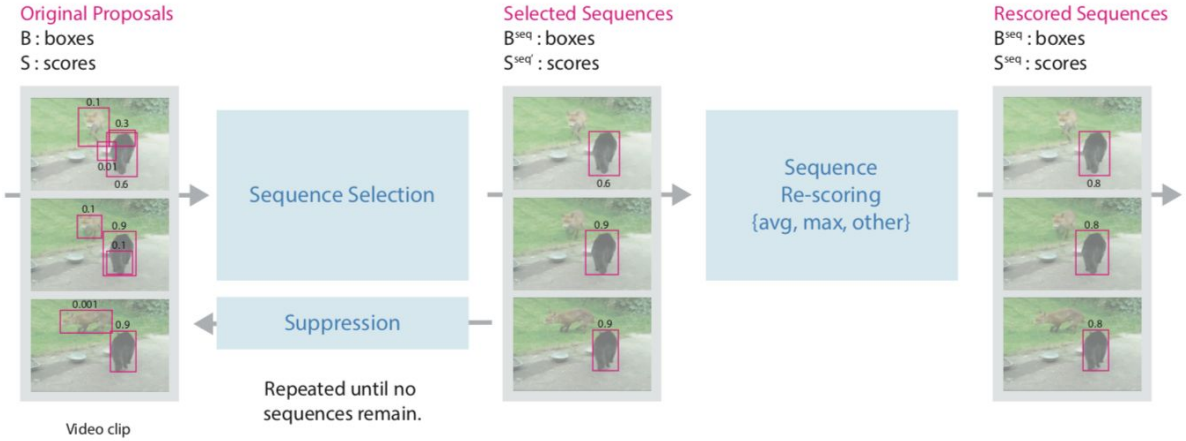
une séquence d’une vidéo donné :

1. Les propositions des régions
2. Les notes correspondantes de la classification

notre méthode :

associer des bounding boxes dans les frames adjacents selon un critère de superposition simple.

La réalisation:
Seq-NMS for Video object Detection



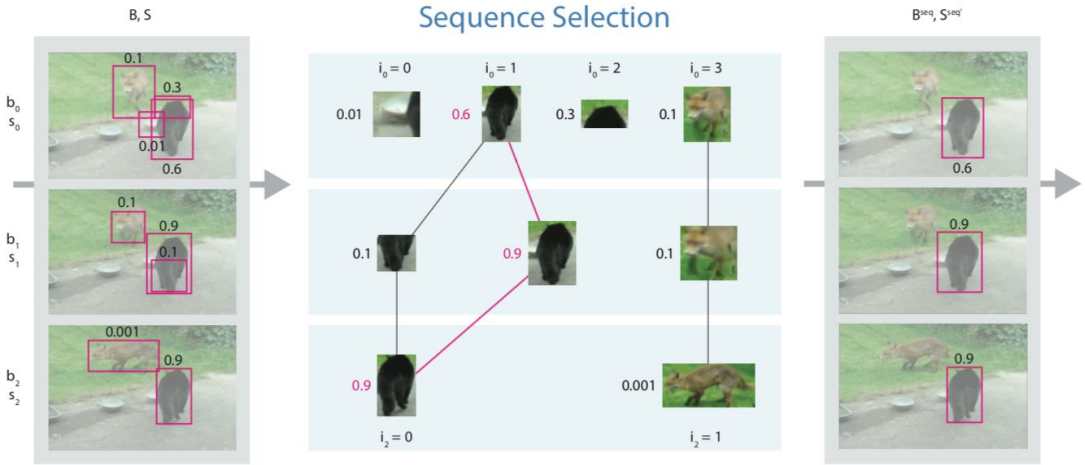
Seq-NMS a trois étapes principales :

(1) Sélection de séquence

(2) Renouveler des notes de séquence

(3) Suppression

La réalisation:
Seq-NMS for Video object Detection

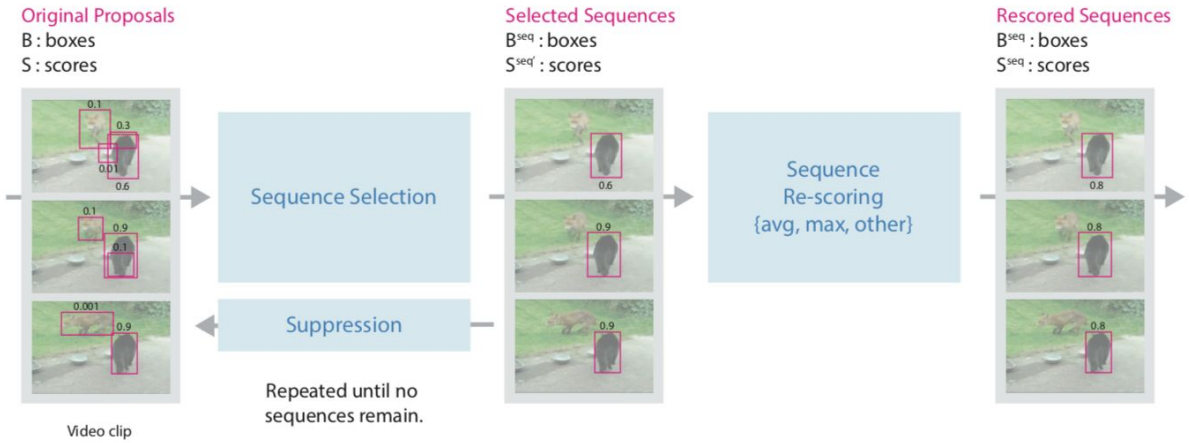


1.Sélection de séquence

la note qui plus élevée

Supprime les boxes qui sont des superpositions grandes

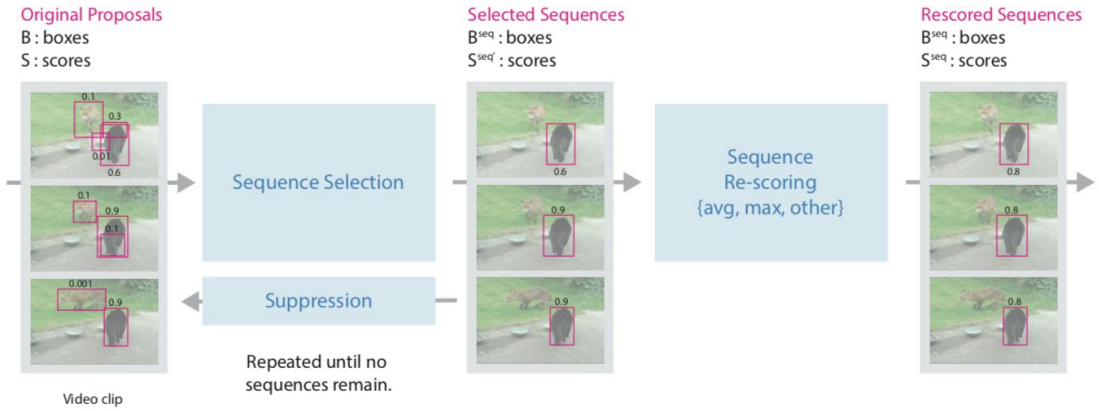
La réalisation:
Seq-NMS for Video object Detection



2.Renouveler des notes de séquence

Appliquer une fonction visant à obtenir une nouvelle note

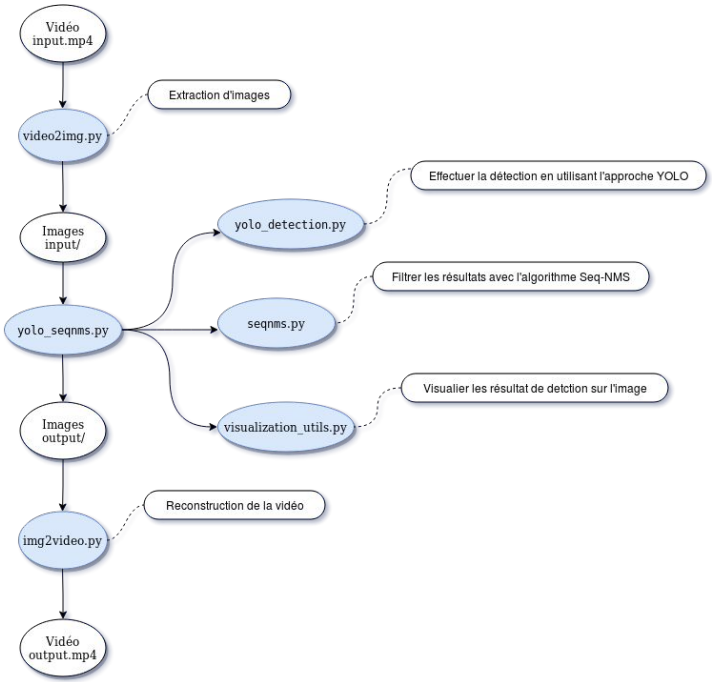
La réalisation:
Seq-NMS for Video object Detection



3.Suppression

Supprimer tous les boxes dans la même frame qui ont les superpositions suffisantes.

La réalisation: Architecture du système

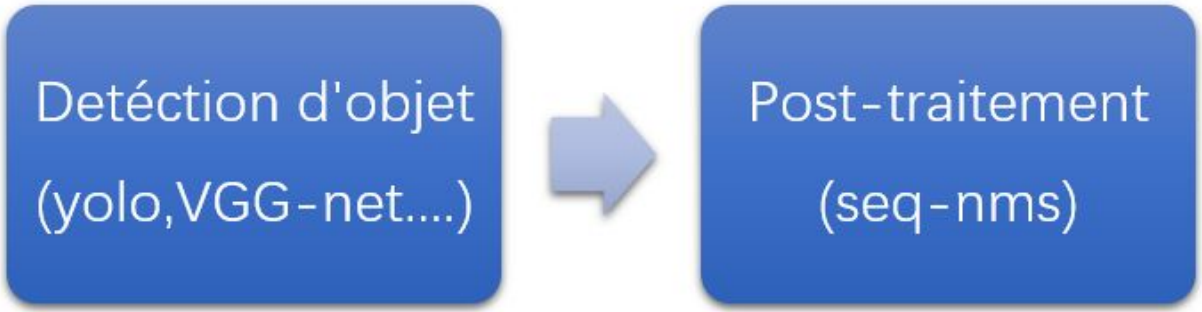


Le code source est publié sur github.

Bilan : Atouts et Limites

Atouts

1. Bon implémentation



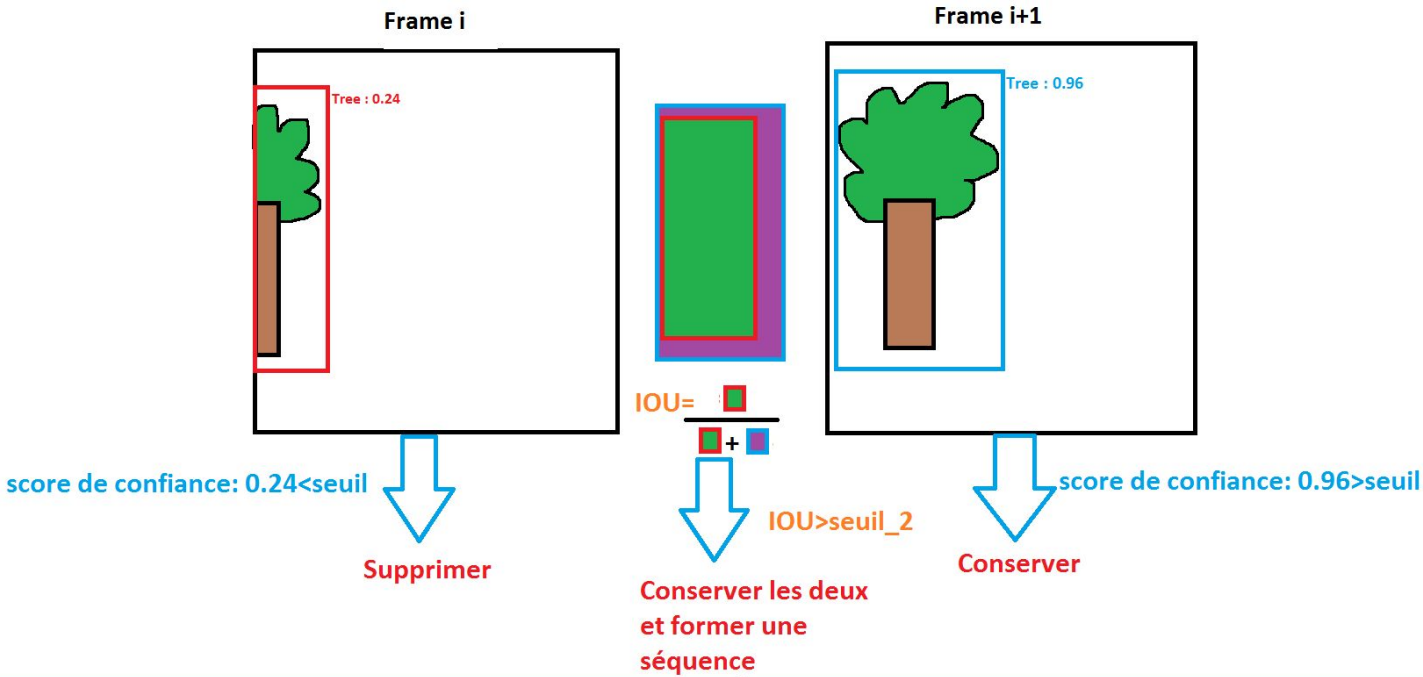
Atouts

2. Correction des problèmes lors d’un suivi

Les problèmes : occlusion,flou, changement drastique de taille d’objet

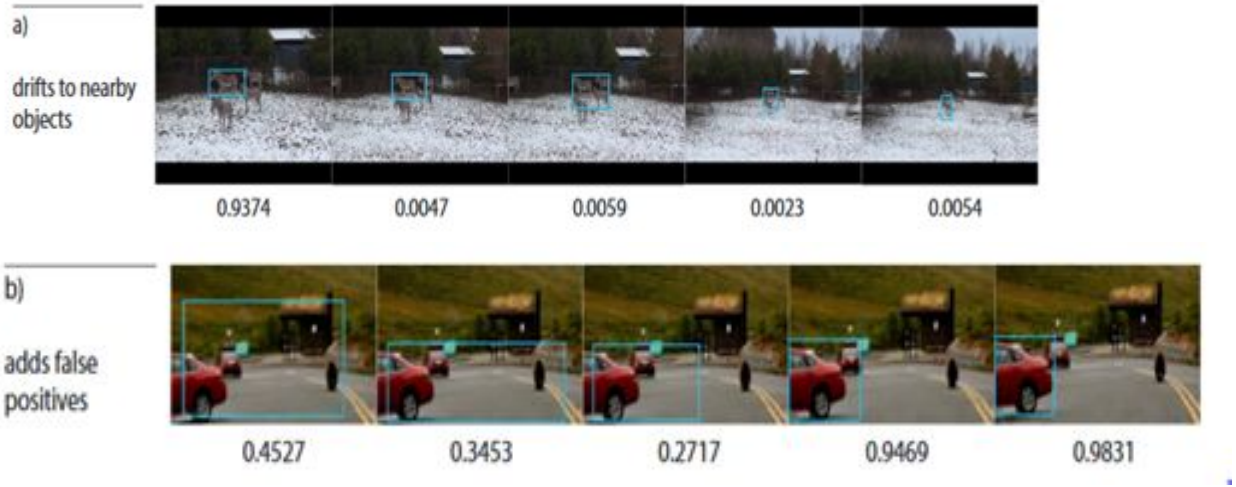
Score faible de confiance

Exemple: occlusion



Limites

1. **Non en temps réel**
2. **Suivi dégradé en certains cas**



Démonstration

Une clip vidéo de <**Kingsman**>

Merci de votre attention