UQAC

Université du Québec
à Chicoutimi

DÉPARTEMENT DES SCIENCES APPLIQUÉES

MODULE D'INGÉNIERIE

Cours : Intelligence artificielle et reconnaissance Des formes (6GEI608)

Présenté : M. louis-André Guérin

Par AICHA KABA

Table de matière

1. Introduction

Notre étude se concentre sur la complexité du suivi d'objets dans les vidéos et l'importance d'utiliser des techniques et des algorithmes avancés. Notre principal objectif est de créer un algorithme de suivi d'objets dans les vidéos enregistrées par une caméra fixe, qui permet de détecter et de segmenter les objets en mouvement dans la séquence. Nous comparons deux techniques de détection de mouvement : la détection en différenciant l'image de fond et les images de la séquence, et la détection en différenciant les images intermédiaires.

Pour chaque image, nous analysons afin de détecter un contour initial qui englobe tous les objets d'intérêt. Dans les prochaines sections, nous détaillerons les différentes étapes de réalisation, les algorithmes utilisés, et les problèmes rencontrés lors de la réalisation de ce projet.

2. Raisonnement du projet

Dans ce projet, nous avons principalement travaillé sur la détection des mouvements dans une vidéo capturée par une caméra fixe. Pour ce faire, nous avons commencé par lier la vidéo capturée et la convertir en noir et blanc **(voir la figure 1).** Ensuite, nous avons appliqué un flou gaussien en utilisant un kernel. Nous avons créé une fonction qui nous permet d'augmenter ou de diminuer ce flou, puis nous avons affiché la vidéo avec ce fond **(voir la figure 3).** Par la suite, nous avons fait la différence entre la vidéo originale et

celle avec le flou **(mask**) **(voir la figure 2).** Enfin, nous avons créé une fonction qui permet de dessiner les contours **(voir la figure 4).**

```
kernel_blur=5
seuil=15
surface=5000
ret, originale=cap.read()
originale=cv2.cvtColor(originale, cv2.COLOR_BGR2GRAY)
originale=cv2.GaussianBlur(originale, ksize: (kernel_blur, kernel_blur), sigmaX: 0)
kernel_dilate=np.ones( shape: (5, 5), np.uint8)
```

Figure 1: conversion d'image en noir blanc

```
ret, frame=cap.read()
gray=cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
gray=cv2.GaussianBlur(gray, ksize: (kernel_blur, kernel_blur), sigmaX: 0)
ici je fait la difference entre image recupé et celle avec le floue
mask=cv2.absdiff(originale, gray)
# pour detecter les objets
```

Figure 2: différence entre image original et floutée

```
# ici cest augmenter l'intensité du flou
mask=cv2.threshold(mask, seuil, maxval: 255, cv2.THRESH_BINARY)[1]
#cv2.imshow("frame", mask)
mask=cv2.dilate(mask, kernel_dilate, iterations=3)
contours, nada=cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
frame_contour=frame.copy()
```

Figure 3: augmenter ou diminuer l'intensité du floue

```
contours, nada=cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
frame_contour=frame.copy()
# cette fonction permet de dessiner mes contours
for c in contours:
    cv2.drawContours(frame_contour, contours: [c], contourIdx: 0, color: (0, 255, 0), thickness: 5)
    # ici je calcul le contour et selectionne la surface
    if cv2.contourArea(c)<surface:
        continue
        # pour entoure le mouvement avec le rectangle
    x, y, w, h=cv2.boundingRect(c)
    cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 0, 255), 2)
```

Figure 4 pour dessiner le contour

3. Algorithmes et technologies utilisés

Plusieurs méthodes peuvent être utilisées pour détecter des objets. Parmi ces méthodes, on trouve le détecteur MultiBox single shot (SSD), l'algorithme de différence d'image, l'algorithme de soustraction d'arrière-plan, l'algorithme de flux optique et l'algorithme d'apprentissage automatique. Parmi ceux-ci, nous avons choisi l'algorithme de différence d'image. Ce dernier nous permet de comparer les images successives capturées par la caméra en calculant la différence de pixels entre elles. Si cette différence dépasse un certain seuil prédéfini, cela signifie qu'il y a un mouvement.

Nous avons également utilisé l'algorithme de soustraction d'arrière-plan. Dans celui-ci, une image de référence de l'arrière-plan sans aucun mouvement est capturée au préalable. Ensuite, chaque image est soustraite de cette image de référence afin de détecter les changements et les mouvements.

Pour calculer la direction et l'amplitude du déplacement d'image, nous avons utilisé l'algorithme de flux optique. Cela nous a permis de détecter les mouvements dans le flux vidéo.

En ce qui concerne l'apprentissage automatique, nous avons utilisé des modèles tels que les réseaux de neurones convolutifs (CNN) pour la détection de mouvement. Ces modèles nous ont permis de former de vastes ensembles de données pour identifier et classer les événements de mouvement.

En termes de technologies, nous avons utilisé des bibliothèques Python comme Open CV, qui offre des fonctionnalités avancées pour la détection de mouvement. **Voir la figure ci-dessous.**

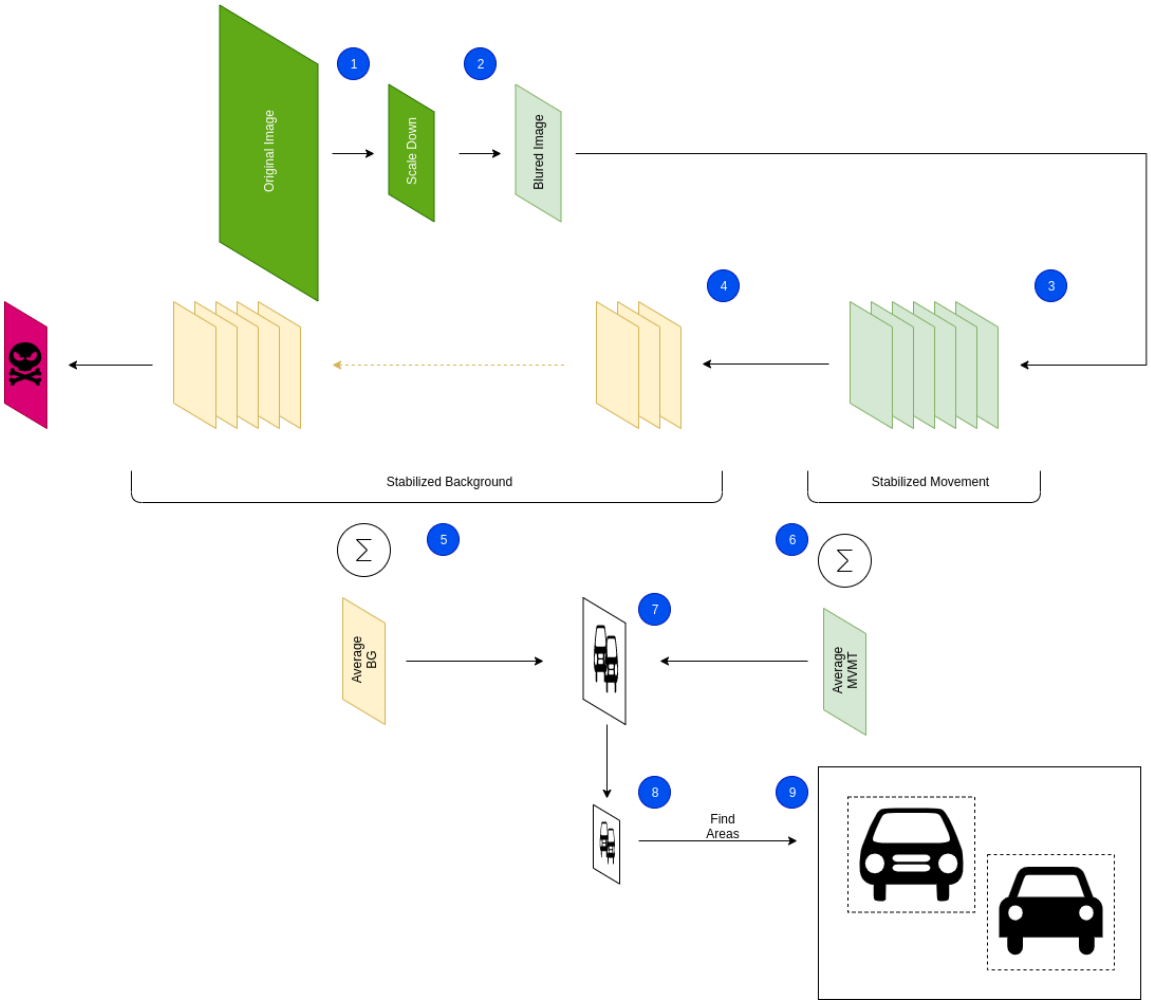


Figure 5: exemple de processeur pour la détection d'image

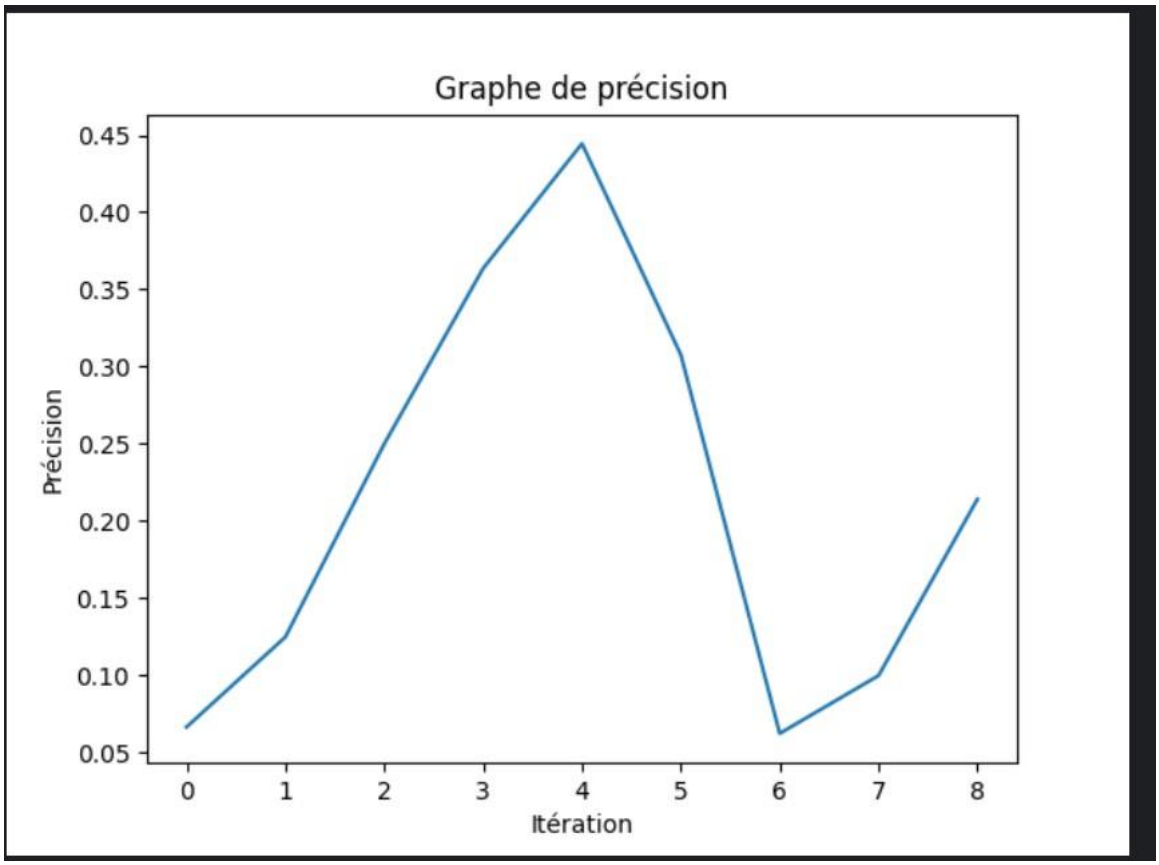
Cette illustration met en évidence les divers phénomènes naturels environnant notre vidéo, capturée par la caméra juste avant de détecter un mouvement.

4. Explication détaillée du résultat

Dans le cadre de ce projet, nous utilisons la bibliothèque **OpenCV** pour capturer une vidéo et appliquer des transformations d'images afin de détecter les mouvements dans cette vidéo. Tout d'abord, nous avons commencé par initialiser la webcam, puis nous avons défini certains paramètres tels que la taille du flou **(kernet_blur),** le seuil de détection des différences entre les images **(seuil)** et la surface minimale des objets détectés. Nous avons ensuite converti l'image de la vidéo en niveaux de gris et appliqué un flou gaussien avant de l'afficher avec cette commande **cv2.imshow("mask", mask).** Ensuite, nous avons utilisé le seuil de différence pour créer un masque binaire des régions où des mouvements ont été détectés. Ce masque a été dilaté et les contours des objets

détectés ont été trouvés. Ces contours ont été dessinés sur l'image d'origine et un rectangle a été ajouté autour des objets détectés s'ils dépassent une certaine surface minimale. Enfin, nous avons affiché l'image avec les contours grâce à **cv2.imshow("contour", frame_contour).**

Les graphiques ci-dessous (Figure 6 et 7) présentent la précision de chaque détection de mouvement.



.

Figure 6: graphe de précision

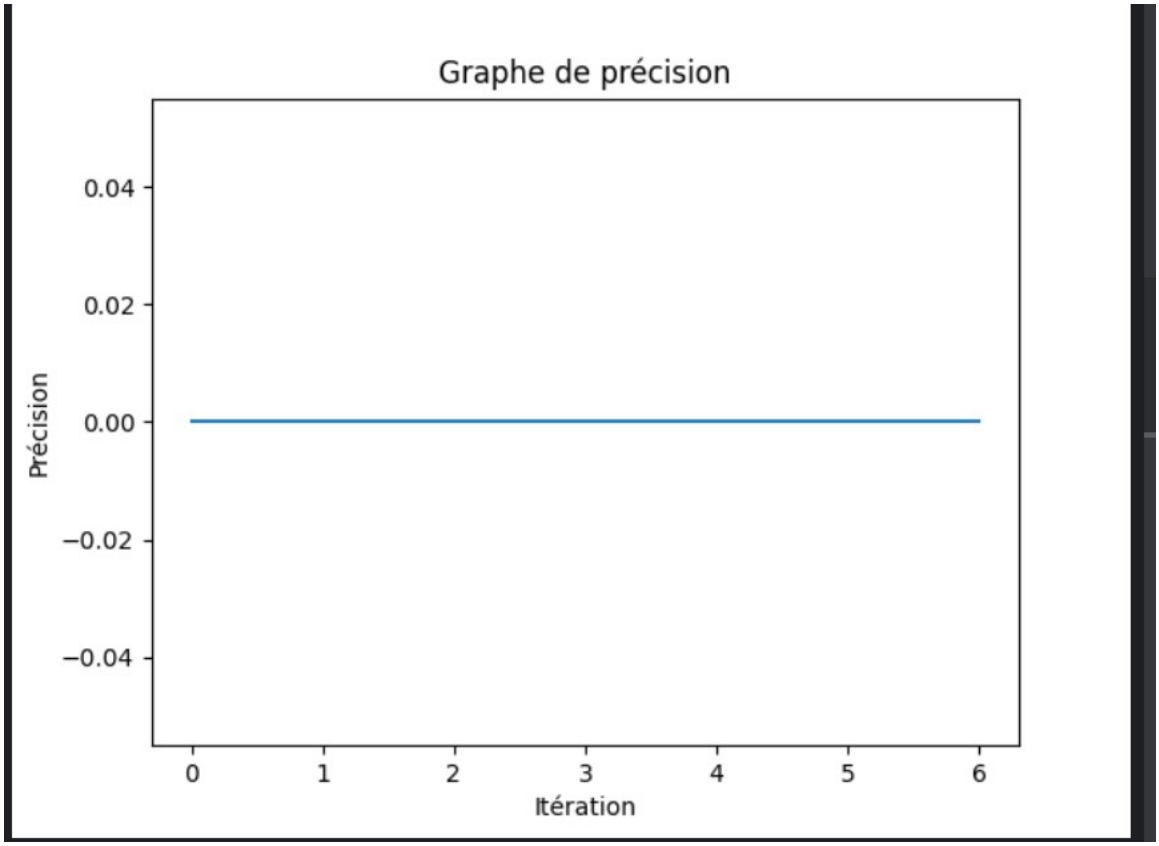


Figure 7 : graphe de précision

Le schéma présenté ci-dessous illustre notre matrice de confusion, mise à jour à chaque détection de mouvement et ce, jusqu'à la fin de la vidéo.

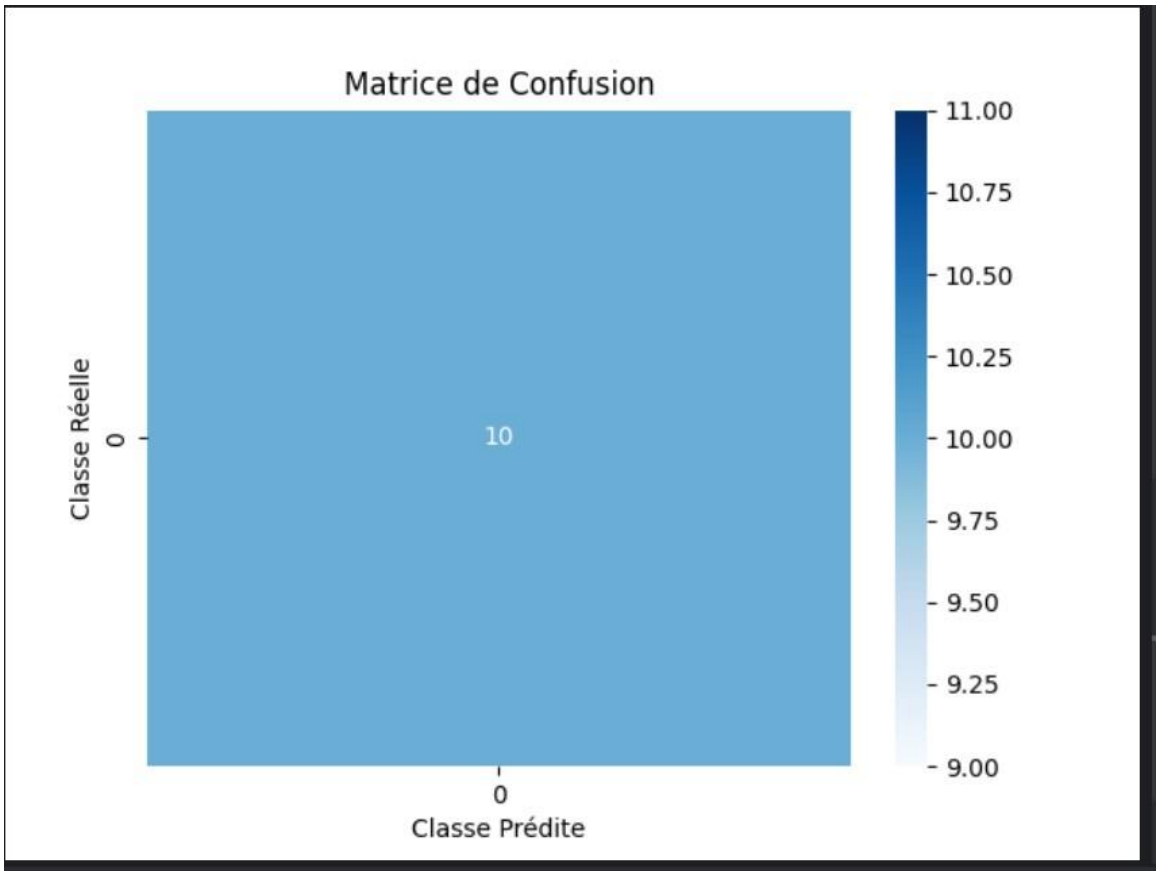


Figure 8: matrice de confusion

La figure ci-dessous montre le taux d'apprentissage de notre modelé

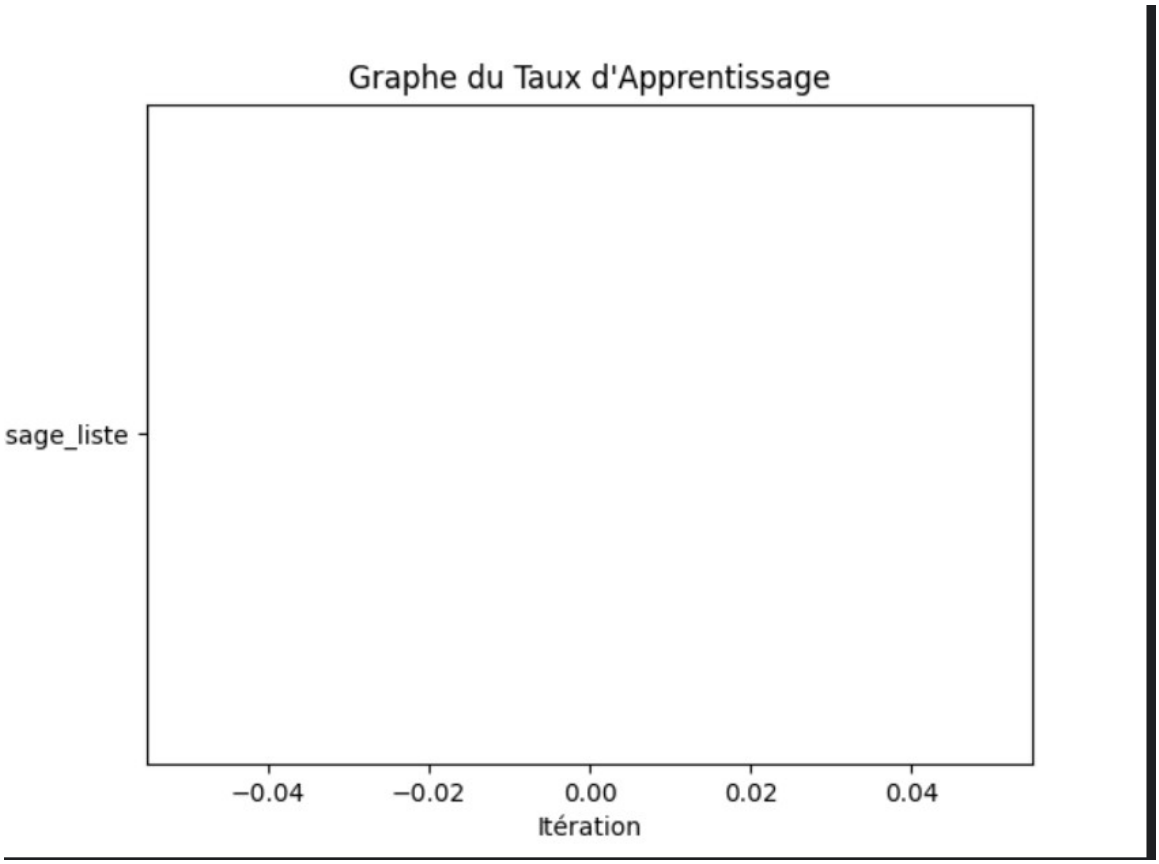


Figure 9: le taux d'apprentissage du modèle

5. Méthodologie adoptée

Dans ce projet de détection de mouvement, nous avons suivi plusieurs étapes afin de proposer une solution. Tout d'abord, nous avons capturé une vidéo à partir d'une source webcam. Avant de procéder à la détection, nous avons prétraité les images de la vidéo pour améliorer leur qualité et faciliter la détection. Une fois le prétraitement terminé, nous avons calculé le fond (reconnaissance des formes).

Après avoir calculé le fond, nous avons soustrait chaque image de la vidéo à ce fond pour détecter les objets en mouvement. Cette étape a permis d'obtenir une image binaire dans laquelle les pixels appartenant aux objets en mouvement sont mis à 1, tandis que les pixels correspondant au fond sont mis à 0. Pour détecter le mouvement, nous avons utilisé la fonction **drawcoutours d'OpenCV** afin de trouver les contours des objets en mouvement (déplacement et classification).

Une fois les contours détectés, nous avons suivi les objets en mouvement en calculant leur trajectoire à partir des positions des contours dans chaque image(localisation).

6. Problème rencontré

Dans le cadre de ce projet, plusieurs défis peuvent survenir. Par exemple, les variations d'éclairage dans la scène peuvent compliquer la distinction entre les objets en mouvement et les changements d'éclairage, entraînant ainsi des détections de mouvement incorrectes. Les ombres, lorsqu'elles sont présentes, peuvent être confondues avec des objets en mouvement, générant des détections erronées. Distinguer les ombres des objets en mouvement requiert généralement une analyse approfondie. De plus, la rapidité des déplacements d'objets en mouvement peut poser des difficultés à l'algorithme de détection, entraînant des détections imprécises ou la perte de suivi de l'objet.

L'efficacité du réseau est cruciale pour une détection de mouvement fiable. Un réseau peu fiable risque de produire des fausses alertes en signalant un mouvement qui n'a pas réellement eu lieu ou en négligeant un mouvement réel, pouvant ainsi causer des perturbations inutiles ou, au contraire, ne pas réagir lorsqu'une détection est nécessaire.

Pour résoudre ces problèmes, il est souvent nécessaire de recourir à des techniques avancées de traitement d'image et à des algorithmes plus complexes.

7. Conclusion

Ce projet était principalement centré sur la détection de mouvement dans une vidéo capturée par une caméra fixe, avec une application basée sur l'intelligence artificielle. À la conclusion de cette initiative, nous avons réussi à détecter le mouvement en utilisant diverses méthodes et algorithmes. Cependant, des défis sont survenus lorsque la qualité de la vidéo capturée était insuffisante, rendant la détection du mouvement difficile sans recourir à des algorithmes plus avancés. Ce projet nous a offert l'occasion d'explorer les avantages et les limites de chaque méthode.

En réponse à vos conseils prodigués lors de la présentation, nous avons pris une nouvelle vidéo. Du point de vue de la précision, nous observons une courbe plus significative, illustrant une amélioration dans les résultats obtenus.

8. Référence

Détection d'objets à l'aide de Single Shot MultiBox Detection (SSD) et du réseau de neurones profonds (DNN) d'OpenCV (ichi.pro)

Détection de pose et de mouvement en Python expliquée avec des projets (ichi.pro)

Détection de mouvement OpenCV | Delft Stack

OpenCV – Détection de Mouvement – e-techno-tutos

PenséeArtificielle.(2018).Focus : MobileNet-SSD.Repéré à: https://penseeartificielle.fr/mobilenet-ssd-identifier-objets-camera-smartphone/