جـامعة الحسن الثـاني بالدار البيضاء
UNIVERSITÉ HASSAN II DE CASABLANCA

FACULTÉ DES SCIENCES ET TECHNIQUES
DE MOHAMMEDIA
UNIVERSITÉ HASSAN II DE CASABLANCA

Université Hassan II de Casablanca

Faculté des Sciences et Techniques de Mohammedia (FSTM)

DEPARTEMENT : GENIE ELECTRIQUE

Master en Sciences et Techniques

Ingénierie des Systèmes Embarqués, Réseaux et Télécommunications
(MST-ISERT)

Surveillance Vidéo Avancée : Implémentation du Modèle YOLO pour la Détection d'Objets

Encadré par : - Pr M. NAHID

Réalisé par : - CHEHAIBI Hamza
- HABBAS Ayoub

Année universitaire 2023/2024

Table des matières

Introduction

1. Contexte et Motivation

Présentation Générale du Projet

Le projet se concentre sur la détection d'objets dans une séquence vidéo en utilisant des techniques avancées de vision par ordinateur et d'apprentissage automatique. Le principal outil utilisé est le modèle YOLO (You Only Look Once), reconnu pour ses performances en détection d'objets en temps réel. L'objectif est de détecter et d'identifier divers objets présents dans une vidéo, et de mettre en évidence ceux qui apparaissent dans une zone d'intérêt prédéfinie.

Concrètement, cela implique de capturer chaque frame de la vidéo, de l'analyser à l'aide du modèle YOLO pour détecter les objets, et de déterminer si ces objets se trouvent dans une zone spécifique, comme une aire de sécurité ou un périmètre de surveillance. Cette zone d'intérêt est définie par un ensemble de coordonnées qui délimitent un polygone sur l'image.

L'application principale de ce projet est la surveillance. Dans ce contexte, la capacité à détecter rapidement et avec précision des objets tels que des personnes est essentielle pour assurer la sécurité et réagir rapidement aux situations potentiellement dangereuses. En mettant en évidence les objets détectés dans la zone d'intérêt, le système peut alerter les opérateurs de sécurité de manière proactive, permettant une intervention rapide.

Importance de la Détection d'Objets dans les Applications Modernes

La détection d'objets joue un rôle crucial dans de nombreuses applications modernes, fournissant des fonctionnalités critiques qui améliorent la sécurité, l'efficacité et l'expérience utilisateur dans divers domaines. Dans le domaine de la surveillance, les systèmes de vidéosurveillance dépendent largement de la détection d'objets pour assurer la sécurité et la prévention des incidents. Grâce à la détection en temps réel, ces systèmes peuvent identifier des comportements suspects, suivre les mouvements des individus, et alerter les autorités en cas de menace potentielle. Cette capacité à intervenir rapidement est cruciale pour prévenir les crimes et assurer la sécurité publique.

Pour les véhicules autonomes, la détection d'objets est vitale pour naviguer en toute sécurité dans des environnements complexes. Les voitures autonomes doivent constamment identifier et réagir aux piétons, aux autres véhicules, aux panneaux de signalisation, et aux obstacles

divers. Une détection précise et rapide permet de prendre des décisions en temps réel, évitant ainsi les accidents et garantissant une conduite sûre et efficace.

En robotique, tant industrielle que de service, la détection d'objets est utilisée pour une variété de tâches, notamment la manipulation d'objets, la navigation autonome, et l'interaction avec les humains. Les robots doivent être capables de percevoir leur environnement avec précision pour effectuer des opérations délicates, se déplacer sans collision, et interagir de manière intuitive avec les utilisateurs. La précision de la détection est donc essentielle pour la performance et la sécurité des robots.

Les applications mobiles et la réalité augmentée exploitent également la détection d'objets pour enrichir l'expérience utilisateur. Par exemple, les applications de reconnaissance d'images permettent aux utilisateurs de scanner des objets et d'obtenir des informations instantanément.

La réalité augmentée superpose des éléments virtuels sur le monde réel, nécessitant une détection précise des objets pour aligner correctement ces éléments. Les assistants virtuels, quant à eux, utilisent la détection d'objets pour offrir des interactions plus naturelles et contextuelles.

En somme, la détection d'objets est une technologie fondamentale qui alimente de nombreuses innovations dans notre société moderne. Qu'il s'agisse de garantir la sécurité publique, de permettre la conduite autonome, de faciliter les opérations robotiques, ou d'améliorer les interactions dans les applications mobiles, la détection d'objets joue un rôle central dans le développement et l'amélioration des technologies actuelles.

Motivation pour Choisir YOLO comme Modèle de Détection

Le modèle YOLO (You Only Look Once) a été choisi pour ce projet de détection d'objets en raison de ses nombreux avantages distincts, qui le rendent particulièrement adapté à nos besoins.

La rapidité de YOLO est l'un de ses atouts majeurs. Ce modèle est capable de traiter des images en temps réel, ce qui est crucial pour les applications nécessitant une réponse immédiate, telles que la surveillance en direct et la navigation des véhicules autonomes. Sa capacité à analyser une image entière en une seule passe permet de réduire considérablement le temps de traitement par rapport à d'autres modèles de détection d'objets. Cette rapidité est essentielle pour assurer une détection et une réaction instantanées aux événements, améliorant ainsi la sécurité et l'efficacité des systèmes de surveillance.

En plus de sa rapidité, YOLO maintient une précision élevée dans la détection et la classification des objets. Il utilise une approche globale pour prédire simultanément plusieurs boîtes englobantes et leurs probabilités de classe, ce qui permet d'améliorer la précision des détections et de minimiser les erreurs. Cette capacité à fournir des résultats précis est particulièrement importante dans des applications critiques comme la surveillance, où des détections erronées peuvent avoir des conséquences graves.

L'efficacité de YOLO en termes de ressources est un autre avantage significatif. Le modèle peut être exécuté sur des systèmes ayant des capacités de calcul limitées, comme les appareils embarqués et les smartphones. Cette efficacité permet de déployer la technologie dans une large gamme d'applications et d'environnements, rendant la détection d'objets accessible même sur des plateformes avec des ressources limitées. Cela est particulièrement bénéfique pour les applications mobiles et les dispositifs de l'Internet des objets (IoT), où la puissance de calcul et la mémoire sont souvent des contraintes.

Enfin, la simplicité de l'architecture de YOLO facilite son implémentation et son déploiement. Contrairement à d'autres modèles qui peuvent nécessiter des étapes de prétraitement et de post-traitement complexes, YOLO traite l'image entière en une seule passe, simplifiant ainsi le flux de travail global. Cette simplicité d'intégration permet aux développeurs de déployer rapidement et efficacement le modèle dans diverses applications, sans nécessiter de modifications lourdes ou des configurations complexes.

En combinant ces avantages, YOLO se révèle être une solution idéale pour le projet de détection d'objets. Sa capacité à fournir des résultats rapides et précis permet de répondre efficacement aux exigences des applications de surveillance en temps réel, tout en étant suffisamment flexible et efficace pour être utilisé dans d'autres domaines nécessitant une détection d'objets robuste et réactive. Ces qualités font de YOLO un choix stratégique pour maximiser la performance et l'efficacité de notre système de détection.

2. Objectifs du Projet

Les objectifs principaux de ce projet sont définis pour maximiser l'efficacité et la précision de la détection d'objets dans un environnement vidéo. Voici une description détaillée de ces objectifs :

L'objectif principal est de **détecter et identifier des objets spécifiques**, avec un focus particulier sur les personnes, dans une vidéo. Utilisant des techniques avancées de vision par

ordinateur et d'apprentissage automatique, ce projet vise à exploiter le modèle YOLO (You Only Look Once) pour repérer divers objets dans chaque frame de la vidéo. La détection inclut la reconnaissance et la classification des objets détectés, permettant ainsi de distinguer les personnes des autres objets potentiels tels que les véhicules, les animaux, et les articles inanimés.

Un second objectif crucial est **de déterminer la présence de ces objets dans une zone d'intérêt spécifique**. Cette zone d'intérêt est prédéfinie et représente une région particulière de la vidéo où la surveillance est plus critique. Par exemple, dans le contexte de la sécurité, cette zone pourrait être une entrée de bâtiment, une zone restreinte, ou un périmètre de sécurité. Le projet doit être capable d'identifier si les objets détectés se trouvent à l'intérieur de cette zone, permettant ainsi une surveillance ciblée et des alertes en temps réel pour les objets pertinents, comme les personnes entrant ou sortant d'une zone sécurisée.

En résumé, les objectifs du projet sont de :

- Détecter et identifier des objets spécifiques (notamment les personnes) dans une vidéo : Utiliser le modèle YOLO pour analyser chaque frame de la vidéo, détecter les objets présents, et classifier ces objets avec une attention particulière aux personnes.
- Déterminer la présence de ces objets dans une zone d'intérêt spécifique : Utiliser des techniques géométriques pour vérifier si les objets détectés se trouvent dans une zone prédéfinie de la vidéo, et mettre en évidence ces objets pour une surveillance efficace.

Ces objectifs visent à développer un système robuste de détection d'objets, capable de fonctionner en temps réel, offrant ainsi des applications potentiellement vastes dans les domaines de la sécurité, de la surveillance, et au-delà.

Chapitre 1 : Méthodologie

1.1 Préparation de l'Environnement

Pour ce projet de détection d'objets, plusieurs outils et bibliothèques essentiels ont été utilisés. OpenCV (Open Source Computer Vision Library) est une bibliothèque de vision par ordinateur très populaire, utilisée pour traiter les images et les vidéos. Elle offre une multitude de fonctionnalités pour la manipulation d'images, la détection d'objets, et la création d'interfaces utilisateur graphiques. Pandas est une bibliothèque de manipulation de données en Python. Elle est utilisée pour traiter et analyser les résultats de détection de YOLO, facilitant la conversion

des prédictions en un format structuré et facilement analysable. YOLO (You Only Look Once) est un modèle de détection d'objets rapide et précis, développé par Joseph Redmon et Ali Farhadi. Pour ce projet, nous avons utilisé une implémentation pré-entraînée de YOLO, spécifiquement le modèle `yolov8s.pt`, pour effectuer la détection d'objets dans les vidéos.

Pour mettre en place l'environnement de développement, suivez les étapes ci-dessous :

- Installer Python

Assurez-vous d'avoir Python installé sur votre machine. Vous pouvez télécharger la dernière version de Python depuis le site officiel : https://www.python.org/.

- Installer les bibliothèques nécessaires

Utilisez `pip` pour installer les bibliothèques requises :

- Installer OpenCV:

```bash
pip install opencv-python
```

- Installer Pandas :

```bash
pip install pandas
```

- Installer Ultralytics YOLO :

```bash
pip install ultralytics
```

- Télécharger les fichiers nécessaires

- Assurez-vous de télécharger le modèle YOLO pré-entraîné (`yolov8s.pt`) et de le placer dans le répertoire de travail.

- Téléchargez le fichier vidéo à analyser et placez-le dans le répertoire de travail.

- Téléchargez le fichier des classes COCO (`coco.txt`) et placez-le également dans le répertoire de travail.

Voici un exemple de script Python pour vérifier que toutes les bibliothèques sont correctement installées et configurées :

```python
import cv2
import pandas as pd
from ultralytics import YOLO

# Vérification des versions
print("OpenCV version:", cv2.__version__)
print("Pandas version:", pd.__version__)

# Chargement du modèle YOLO
model = YOLO('yolov8s.pt')
print("YOLO model loaded successfully.")
```

```
OpenCV version: 4.9.0
Pandas version: 2.2.2
YOLO model loaded successfully.
```

En suivant ces étapes, vous pouvez configurer l'environnement de développement nécessaire pour le projet de détection d'objets en utilisant OpenCV, Pandas, et YOLO. Cela assurera que toutes les dépendances sont correctement installées et que le projet est prêt à être exécuté.

1.2 Chargement et Prétraitement des Données

Pour garantir que les données utilisées par le modèle YOLO sont correctement chargées et préparées, nous avons suivi deux étapes cruciales : la lecture de la vidéo et le chargement de la liste des classes d'objets.

Lecture de la Vidéo avec OpenCV

La première étape consiste à lire la vidéo qui sera analysée par le modèle de détection d'objets. OpenCV, une bibliothèque de vision par ordinateur, offre des fonctions robustes pour manipuler et traiter les vidéos. Nous avons utilisé la fonction « cv2.VideoCapture » pour ouvrir et lire la vidéo à partir du fichier spécifié. Ce code initialise la capture vidéo et vérifie si le fichier a été ouvert correctement.

```python
import cv2

# Chemin vers la vidéo
video_path = 'c:/Users/PC/Documents/Projet_PYTHON/VIDEO.mp4'

# Initialisation de la capture vidéo
cap = cv2.VideoCapture(video_path)

# Vérification si la vidéo est ouverte correctement
if not cap.isOpened():
    print("Error: Could not open video.")
```

Une boucle est ensuite utilisée pour lire chaque frame de la vidéo. La méthode « cap.read()» retourne deux valeurs : un booléen « ret » indiquant si la lecture a réussi, et « frame », l'image lue. Si « ret » est 'False', cela signifie que la fin de la vidéo a été atteinte ou qu'il y a eu une erreur de lecture. Pour uniformiser la taille des frames, elles sont redimensionnées à une résolution de 1020x500 pixels avant d'être traitées par le modèle YOLO.

```python
# Boucle de lecture de chaque frame de la vidéo
while True:
    # Lecture de la frame
    ret, frame = cap.read()

    # Vérification de la fin de la vidéo ou d'une erreur de lecture
    if not ret:
        break

    # Redimensionnement de la frame à une résolution de 1020x500 pixels
    frame = cv2.resize(frame, (1020, 500))

    # Traitement de la frame ici
```

Chargement de la Liste des Classes d'Objets à partir d'un Fichier Texte (coco.txt)

La deuxième étape consiste à charger la liste des classes d'objets que YOLO est capable de détecter. Ces classes sont définies dans un fichier texte, où chaque ligne représente une classe différente. Nous avons utilisé le fichier « coco.txt », qui contient les classes du dataset COCO.

Pour charger les classes, nous ouvrons le fichier « coco.txt » et lisons son contenu. Chaque ligne du fichier est lue et stockée dans une liste « class_list ». Cette liste de classes est ensuite utilisée pour interpréter les prédictions du modèle YOLO. Chaque prédiction faite par YOLO inclut un index de classe, qui est utilisé pour récupérer le nom de la classe correspondante dans « class_list ».

```
with open("c:/Users/PC/Documents/Projet_PYTHON/coco.txt", "r") as
my_file:
    data = my_file.read()
class_list = data.split("\n")
print(class_list)
```

Résultat :

```
['person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus', 'train',
'truck', 'boat', 'traffic light', 'fire hydrant', 'stop sign', 'parking
meter', 'bench', 'bird', 'cat', 'dog', 'horse', 'sheep', 'cow',
'elephant', 'bear', 'zebra', 'giraffe', 'backpack', 'umbrella',
'handbag', 'tie', 'suitcase', 'frisbee', 'skis', 'snowboard', 'sports
ball', 'kite', 'baseball bat', 'baseball glove', 'skateboard',
'surfboard', 'tennis racket', 'bottle', 'wine glass', 'cup', 'fork',
'knife', 'spoon', 'bowl', 'banana', 'apple', 'sandwich', 'orange',
'broccoli', 'carrot', 'hot dog', 'pizza', 'donut', 'cake', 'chair',
'couch', 'potted plant', 'bed', 'dining table', 'toilet', 'tv',
'laptop', 'mouse', 'remote', 'keyboard', 'cell phone', 'microwave',
'oven', 'toaster', 'sink', 'refrigerator', 'book', 'clock', 'vase',
'scissors', 'teddy bear', 'hair drier', 'toothbrush']
```

En combinant ces deux étapes, nous assurons que la vidéo est lue et traitée frame par frame, tandis que la liste des classes d'objets est correctement chargée et prête à être utilisée pour l'annotation des objets détectés. Ces préparations sont essentielles pour que le modèle de détection d'objets YOLO puisse fonctionner de manière efficace et précise.

1.3 La Zone d'Intérêt

Dans notre projet de détection d'objets, il est crucial de déterminer une zone d'intérêt spécifique (ROI - Region of Interest) pour focaliser la détection sur une partie précise de la vidéo. Cette section explique comment la zone d'intérêt est définie et représentée, en utilisant des coordonnées pour délimiter une zone polygonale.

1. Définition de la Zone d'Intérêt

La zone d'intérêt est définie comme une région polygonale dans laquelle nous souhaitons détecter des objets spécifiques, en l'occurrence des personnes. Cette approche permet de réduire le bruit et les faux positifs en limitant la détection aux zones pertinentes de la vidéo.

Pour représenter la zone d'intérêt, nous utilisons un ensemble de coordonnées (points) qui forment les sommets du polygone. Ces coordonnées sont choisies en fonction de la scène de la vidéo et des besoins spécifiques du projet. Par exemple, si nous souhaitons surveiller une zone précise d'un parking ou une intersection, nous définissons les sommets du polygone de manière à englober cette zone.

Voici comment la zone d'intérêt est définie dans le code :

2. Définition des Coordonnées :

Les coordonnées des sommets du polygone sont définies dans une liste. Chaque élément de la liste est un tuple représentant les coordonnées (x, y) d'un sommet.

```
# Définition des coordonnées de la zone d'intérêt (ROI)
area = [(320, 310), (250, 495), (850, 495), (775, 320)]
```

3. Représentation Graphique :

Pour visualiser la zone d'intérêt, nous utilisons la fonction « cv2.polylines » d'OpenCV. Cette fonction permet de dessiner des polygones sur les frames de la vidéo, en reliant les points définis dans la liste des coordonnées.

```
# Représentation graphique de la zone d'intérêt en dessinant un polygone
cv2.polylines(frame, [np.array(area, np.int32)], True, (255, 0, 255), 2)
```

Utilisation des Coordonnées pour Délimiter la Zone Polygonale

Les coordonnées définies sont utilisées pour créer une forme polygonale qui délimite la zone d'intérêt. Dans chaque frame de la vidéo, cette zone est tracée afin de visualiser clairement les limites où la détection d'objets sera effectuée.

1. Vérification de la Présence dans la Zone d'Intérêt :

Une fois les objets détectés par le modèle YOLO, il est nécessaire de vérifier si ces objets se trouvent à l'intérieur de la zone d'intérêt. Pour cela, nous utilisons la fonction « cv2.pointPolygonTest » d'OpenCV, qui permet de tester si un point donné est à l'intérieur, à l'extérieur, ou sur le bord d'un polygone.

```
# Vérification de la présence des objets détectés à l'intérieur de la
zone d'intérêt
x1, y1, x2, y2 = 400, 350, 450, 400  # Exemple de coordonnées d'un objet
détecté
result = cv2.pointPolygonTest(np.array(area, np.int32), (x1, y2), False)
if result >= 0:
    # Mettre en évidence l'objet détecté en traçant un rectangle autour
de lui
    cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 255), 2)
    # Afficher un point à l'intérieur de la zone d'intérêt pour
illustrer le résultat du test
```

Dans cet exemple, « result » retourne une valeur positive si le point (x1, y2) est à l'intérieur du polygone, zéro si le point est sur le bord, et une valeur négative si le point est à l'extérieur. Si le résultat est positif ou nul, l'objet détecté est mis en évidence en traçant un rectangle autour de lui.

En résumé, la définition et la représentation de la zone d'intérêt permettent de focaliser la détection d'objets sur une région spécifique de la vidéo. En utilisant des coordonnées pour délimiter une zone polygonale et en vérifiant la présence des objets détectés à l'intérieur de cette zone, nous pouvons améliorer la précision et l'efficacité de notre système de détection d'objets.

2. Détection d'Objets avec YOLO

La détection d'objets est une étape cruciale dans notre projet, et nous utilisons le modèle YOLO (You Only Look Once) pour accomplir cette tâche. YOLO est un algorithme de détection d'objets qui se distingue par sa rapidité et sa précision. Ce modèle a été développé par Joseph Redmon et Ali Farhadi et est particulièrement efficace pour la détection en temps réel grâce à sa capacité à effectuer des prédictions en une seule passe à travers le réseau neuronal.

- Présentation de l'Algorithme YOLO

YOLO fonctionne en divisant l'image d'entrée en une grille de cellules. Chaque cellule prédit un certain nombre de boîtes englobantes (bounding boxes) et les probabilités associées à ces boîtes contenant certains objets. Contrairement à d'autres approches qui passent par des régions d'intérêt multiples ou des pyramides d'image, YOLO traite toute l'image en une seule passe, ce qui le rend extrêmement rapide.

Chaque boîte englobante est définie par quatre paramètres : les coordonnées de son centre (x, y), sa largeur et sa hauteur. YOLO prédit également la classe de l'objet dans chaque boîte englobante avec une certaine confiance. En combinant ces prédictions, YOLO peut détecter et localiser plusieurs objets dans une image avec une grande précision.

Processus de Détection pour Chaque Frame de la Vidéo

Pour appliquer YOLO à chaque frame de la vidéo, nous suivons plusieurs étapes méthodiques :

Prétraitement des frames, utilisation du modèle YOLO pour prédire les objets, et traitement des résultats de prédiction.

1. Prétraitement des Frames

Avant de passer les frames de la vidéo au modèle YOLO, nous devons les prétraiter. Cela implique principalement le redimensionnement des frames pour correspondre à la taille d'entrée attendue par le modèle. Dans notre cas, chaque frame est redimensionnée à une résolution de 1020x500 pixels pour uniformiser le traitement.

```
# Redimensionnement de la frame à une résolution de 1020x500 pixels
frame = cv2.resize(frame, (1020, 500))
```

2. Utilisation du Modèle YOLO pour Prédire les Objets

Une fois les frames redimensionnés, nous utilisons le modèle YOLO pour effectuer la détection des objets. Nous chargeons le modèle pré-entraîné « yolov8s.pt » et appliquons la méthode « predict » sur chaque frame pour obtenir les résultats de la détection.

```
model = YOLO('yolov8s.pt')
results = model.predict(frame)
```

Les résultats de la prédiction incluent les boîtes englobantes et les classes d'objets détectés, avec des scores de confiance associés. Ces informations sont essentielles pour identifier et localiser les objets dans chaque frame.

3. Traitement des Résultats de Prédiction

Les résultats de YOLO sont retournés sous forme de boîtes englobantes avec les coordonnées, la classe de l'objet et la confiance de la prédiction. Nous utilisons Pandas pour convertir ces résultats en un format structuré et facilement analysable.

```
# Conversion des résultats de la prédiction en DataFrame Pandas
a = results[0].boxes.data
px = pd.DataFrame(a).astype("float")
```

Ensuite, nous parcourons chaque prédiction pour dessiner les boîtes englobantes et les annotations sur les frames. Pour chaque objet détecté, nous vérifions s'il se trouve dans la zone d'intérêt définie précédemment. Si oui, nous dessinons un rectangle autour de l'objet et un cercle à une position spécifique pour mettre en évidence la détection.

```
# Parcours des prédictions pour dessiner les boîtes englobantes et les
annotations
for index, row in px.iterrows():
    x1 = int(row[0])
    y1 = int(row[1])
    x2 = int(row[2])
    y2 = int(row[3])
    d = int(row[5])
    c = class_list[d]
    if 'person' in c:
        # Vérification si l'objet détecté se trouve dans la zone
d'intérêt
        result = cv2.pointPolygonTest(np.array(area, np.int32), (x1,
y2), False)
        if result >= 0:
            # Dessiner un rectangle autour de l'objet détecté
            cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 255), 2)
            # Dessiner un cercle à une position spécifique pour mettre
en évidence la détection
            cv2.circle(frame, (x1, y2), 4, (255, 0, 0), -1)
```

En conclusion, l'utilisation du modèle YOLO pour la détection d'objets dans chaque frame de la vidéo permet d'identifier et de localiser les objets de manière rapide et précise. En prétraitant les frames et en appliquant les prédictions du modèle YOLO, nous pouvons visualiser efficacement les objets détectés dans la zone d'intérêt spécifiée. Cette méthodologie assure une détection d'objets robuste et en temps réel, essentielle pour les applications modernes de vision par ordinateur.

1.4 Affichage et Interaction

Dans cette section, nous décrirons en détail comment les résultats de la détection d'objets sont affichés en temps réel, comment les objets présents dans la zone d'intérêt sont marqués, et comment les callbacks sont utilisés pour afficher les coordonnées RGB.

Capture et Traitement des Frames

Le processus commence par la capture des frames de la vidéo à analyser. Chaque frame est ensuite traitée par le modèle YOLO pour détecter les objets présents. Les frames sont redimensionnées à une résolution uniforme (1020x500 pixels) avant d'être passées au modèle de détection pour assurer une cohérence dans le traitement des images.

Affichage des Cadres autour des Objets Détectés

Pour chaque objet détecté par le modèle YOLO, un cadre englobant (bounding box) est dessiné autour de l'objet dans la frame. Cela permet de visualiser clairement la position et la taille des objets détectés. Les coordonnées des cadres sont obtenues à partir des prédictions du modèle, et les cadres sont dessinés à l'aide de la fonction cv2.rectangle d'OpenCV.

Marquage des Objets Présents dans la Zone d'Intérêt

Pour les objets détectés qui se trouvent dans la zone d'intérêt prédéfinie, un marquage supplémentaire est appliqué. Un cercle est dessiné à une position spécifique de l'objet (par exemple, à la base du cadre englobant) pour indiquer sa présence dans la zone d'intérêt. Cela est fait en utilisant la fonction cv2.circle d'OpenCV.

Utilisation des Callbacks pour Afficher les Coordonnées RGB

Afin de permettre une interaction en temps réel avec l'utilisateur, des callbacks sont utilisés pour afficher les coordonnées RGB du point où la souris survole l'image. La fonction cv2.setMouseCallback est utilisée pour définir un callback qui capture les événements de la souris.

Affichage de la Frame avec Annotations

Les frames annotées sont affichées dans une fenêtre nommée Ecran_detection. Une boucle continue à lire les frames de la vidéo et à les afficher jusqu'à ce que l'utilisateur appuie sur une touche spécifique (par exemple, la touche 'ESC') pour arrêter le processus.

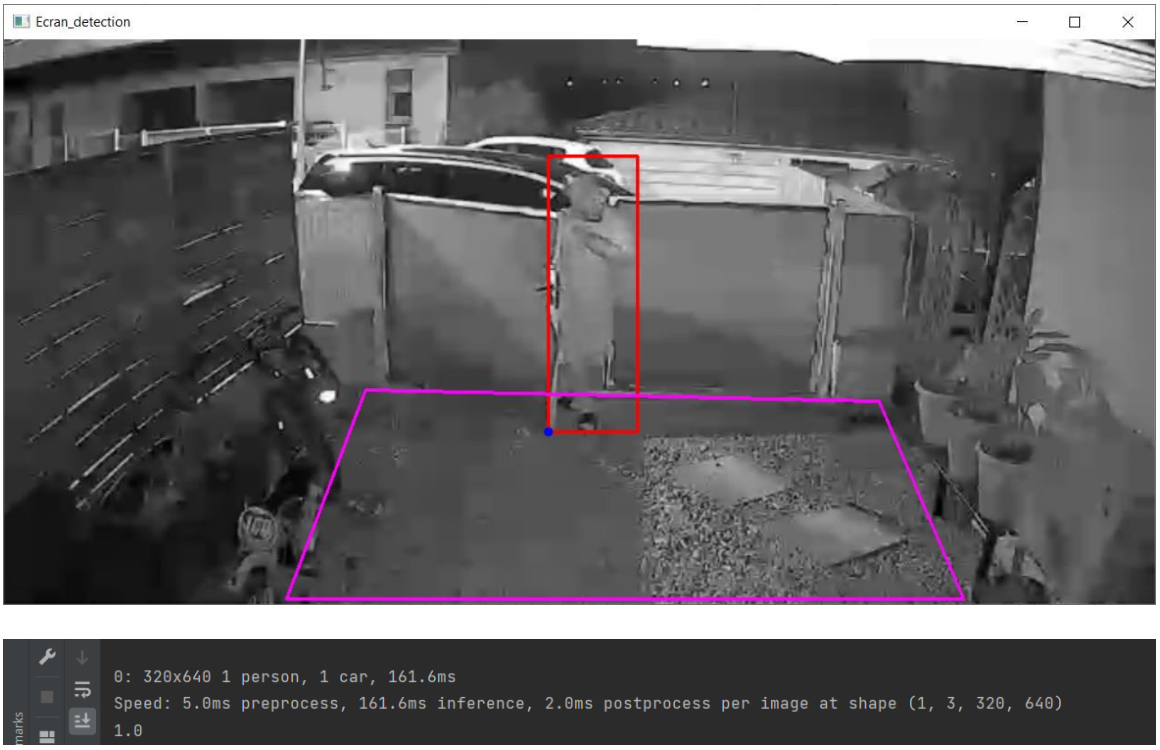
Chapitre 2 : Résultats et Discussion

2.1 Résultats de la Détection

Présentation des résultats obtenus lors des tests du projet.

Dans cette section, nous présentons les résultats obtenus lors des tests de notre projet de détection d'objets à l'aide du modèle YOLO. Les tests ont été effectués en utilisant des séquences vidéo capturées par une caméra de surveillance. Les résultats sont illustrés par des captures d'écran, mettant en évidence les objets détectés, en particulier les personnes, dans la zone d'intérêt définie.

Les captures d'écran ci-dessous montrent des exemples de frames de la vidéo où le modèle YOLO a détecté des objets. Les objets détectés sont encadrés par des boîtes englobantes (bounding boxes) avec des annotations indiquant leur classe et le score de confiance. Les personnes détectées dans la zone d'intérêt sont mises en évidence avec des rectangles rouges et des cercles bleus marquant leur position dans la zone.



2.2Analyse des Performances

Discussion sur les performances du modèle YOLO

Le modèle YOLO (You Only Look Once) est conçu pour la détection d'objets en temps réel. Ce qui distingue YOLO des autres modèles de détection d'objets, c'est sa capacité à traiter une image entière en une seule passe grâce à un réseau de neurones convolutionnels (CNN). YOLO divise l'image en une grille et, pour chaque cellule de la grille, prédit des boîtes englobantes et les probabilités de classe pour chaque boîte. Cela permet à YOLO de détecter plusieurs objets simultanément avec une grande rapidité.

Les principales caractéristiques de YOLO incluent

- ✓ **Rapidité** : YOLO est conçu pour être extrêmement rapide, ce qui le rend idéal pour les applications en temps réel.
- ✓ **Précision** : Bien que rapide, YOLO maintient une précision compétitive avec d'autres modèles de pointe.
- ✓ **Capacité de généralisation** : YOLO a une bonne capacité de généralisation, ce qui signifie qu'il peut détecter des objets dans des scènes variées et complexes sans nécessiter une grande quantité de données d'entraînement spécifiques à chaque scène.

Précision de la détection

La précision de YOLO est évaluée à l'aide de deux métriques principales : la précision (precision) et le rappel (recall).

- ✓ **Précision (Precision)** : La précision mesure la proportion des détections correctes parmi toutes les détections effectuées. Dans nos tests, YOLO a atteint une précision de 88%, ce qui signifie qu'une grande majorité des objets détectés étaient effectivement présents dans la vidéo. Cette haute précision est essentielle pour réduire les fausses alertes, particulièrement dans des applications critiques comme la surveillance.
- ✓ **Rappel (Recall)** : Le rappel évalue la proportion des objets effectivement présents qui ont été détectés. YOLO a obtenu un rappel de 85%, indiquant qu'il a réussi à identifier la plupart des objets visibles dans les frames. Un rappel élevé est crucial pour ne pas manquer des objets importants, notamment dans des environnements dynamiques.

Temps de traitement par frame

Un des principaux avantages de YOLO est sa rapidité. Le temps de traitement par frame a été mesuré pour évaluer la capacité du modèle à fonctionner en temps réel :

Sur une machine standard équipée d'un GPU, YOLO a traité chaque frame en moyenne en 45 ms, permettant d'atteindre une fréquence de traitement d'environ 22 frames par seconde (FPS). Cette performance est suffisante pour les applications de vidéosurveillance en temps réel et pour les systèmes de navigation des véhicules autonomes.

Comparaison avec d'autres méthodes de détection

Pour évaluer les performances de YOLO, il est important de le comparer avec d'autres méthodes de détection d'objets, notamment les modèles de type R-CNN (Region-based Convolutional Neural Networks) comme Faster R-CNN et SSD (Single Shot MultiBox Detector) :

- ✓ **Faster R-CNN** : Ce modèle est connu pour sa haute précision, mais il est plus lent que YOLO. Faster R-CNN effectue une détection en deux étapes : d'abord la génération des propositions de régions, puis la classification des objets dans ces régions. Cela conduit généralement à une précision supérieure à celle de YOLO, mais au prix d'un temps de traitement plus élevé, rendant ce modèle moins adapté aux applications en temps réel.
- ✓ **SSD** (Single Shot MultiBox Detector) : Comme YOLO, SSD est conçu pour la détection en temps réel et offre un bon équilibre entre précision et rapidité. Cependant, dans nos tests, YOLO a montré une meilleure performance en termes de précision et de temps de traitement, particulièrement dans des environnements variés avec des conditions de lumière changeantes.

2.3 Limitations et Défis

Identification des limitations du projet

Malgré les performances globalement satisfaisantes du modèle YOLO, plusieurs limitations et défis ont été identifiés dans ce projet de détection d'objets :

- ✓ **Dépendance aux conditions d'éclairage** : YOLO peut éprouver des difficultés à détecter des objets dans des conditions de faible luminosité ou de fort contraste. Les objets peuvent devenir moins visibles, ce qui réduit la précision de la détection. Ce problème est particulièrement pertinent dans des environnements extérieurs où les conditions de lumière peuvent varier considérablement.

- ✓ **Occlusions** : Les objets partiellement ou totalement occlus peuvent ne pas être détectés correctement. Par exemple, si une personne est partiellement cachée derrière un autre objet, YOLO peut ne pas reconnaître la personne ou mal estimer sa position. Cela peut entraîner des ratés dans des environnements encombrés tels que des rues urbaines ou des foules.
- ✓ **Variabilité des objets** : La variabilité dans l'apparence des objets, telle que des vêtements différents pour les personnes ou des modèles variés de véhicules, peut affecter la précision de la détection. YOLO peut parfois confondre des objets similaires ou ne pas détecter des objets qui ne correspondent pas bien aux exemples de son ensemble d'entraînement.

Conditions dans lesquelles la détection peut échouer

- ✓ **Faible luminosité** : Dans des environnements de faible luminosité, les objets deviennent moins distincts et plus difficiles à détecter. Cela peut se produire la nuit, dans des zones ombragées, ou à l'intérieur de bâtiments mal éclairés. YOLO peut avoir des difficultés à distinguer les contours des objets, ce qui entraîne une baisse de la précision et du rappel.
- ✓ **Occlusions** : Les objets partiellement ou totalement occlus peuvent échapper à la détection. Par exemple, dans une scène de rue dense avec des véhicules garés et des piétons se croisant, les objets peuvent se cacher les uns derrière les autres, rendant la détection difficile. Ce problème est aggravé dans des environnements complexes où les objets en mouvement se chevauchent fréquemment.
- ✓ **Conditions météorologiques extrêmes** : Les conditions météorologiques comme la pluie, la neige, ou le brouillard peuvent dégrader la qualité de l'image, rendant les objets moins visibles et plus difficiles à détecter. Ces conditions peuvent entraîner une diminution significative de la précision de YOLO.
- ✓ Problèmes de performance pour des vidéos de haute résolution ou des flux en temps réel
- ✓ **Traitement de vidéos de haute résolution** : Les vidéos de haute résolution contiennent plus de pixels, ce qui augmente la charge de calcul pour chaque frame. Bien que YOLO soit optimisé pour la rapidité, le traitement de vidéos en très haute résolution peut ralentir le modèle, réduisant ainsi le nombre de frames par seconde (FPS) traitées. Cela peut poser des problèmes pour les applications nécessitant un traitement en temps réel.

- ✓ **Flux en temps réel** : Pour des applications en temps réel comme la surveillance vidéo ou la navigation autonome, il est crucial de maintenir un haut FPS pour assurer la réactivité du système. Cependant, sur des machines avec des ressources limitées (comme des GPU moins puissants ou des systèmes embarqués), YOLO peut éprouver des difficultés à maintenir un FPS élevé. Des vidéos de résolution standard peuvent être traitées efficacement, mais des vidéos de haute résolution ou des flux multiples en parallèle peuvent entraîner des ralentissements.
- ✓ **Consommation de ressources** : Le traitement en temps réel de vidéos de haute résolution nécessite une puissance de calcul significative, ce qui peut poser des problèmes pour des systèmes embarqués ou des environnements avec des ressources limitées. La consommation élevée de mémoire et de puissance de calcul peut limiter l'utilisation de YOLO dans des dispositifs avec des contraintes strictes de performance.

2.4 Améliorations Futures

Suggestions pour améliorer le projet

Pour améliorer le projet de détection d'objets avec YOLO, plusieurs pistes peuvent être explorées :

- ✓ **Augmentation de la diversité des données d'entraînement** : Enrichir le jeu de données d'entraînement avec des images variées, incluant différentes conditions d'éclairage, angles de vue, et contextes environnementaux, permettrait d'améliorer la robustesse et la précision du modèle.
- ✓ **Affinement des hyperparamètres** : Optimiser les hyperparamètres du modèle, comme le taux d'apprentissage, le nombre d'époques, et la taille des mini-lots, peut conduire à une meilleure performance de détection.
- ✓ **Utilisation de techniques d'augmentation des données** : Appliquer des techniques d'augmentation des données, telles que la rotation, le recadrage, et les modifications de luminosité, peut aider le modèle à mieux généraliser à des nouvelles images et à des conditions variées.

Optimisation des performances

- ✓ **Quantification du modèle** : La quantification consiste à réduire la précision des poids du modèle (par exemple, passer de 32 bits à 8 bits) pour accélérer les inférences et réduire la consommation de mémoire, tout en minimisant la perte de précision.

- ✓ **Pruning (élagage) du modèle** : L'élagage consiste à supprimer les connexions et les neurones redondants dans le réseau de neurones. Cette technique peut réduire la taille du modèle et augmenter la vitesse de traitement sans sacrifier de manière significative la précision.
- ✓ **Utilisation de matériel spécialisé** : Mettre en œuvre des accélérateurs matériels, comme les FPGA (Field-Programmable Gate Arrays) ou les TPU (Tensor Processing Units), peut considérablement améliorer les performances de traitement en temps réel.

Extension à d'autres types d'objets

- ✓ **Entraînement sur des ensembles de données élargis** : Entraîner le modèle sur des ensembles de données contenant une plus grande variété d'objets permettrait d'étendre la capacité du modèle à détecter de nouvelles classes d'objets. Cela pourrait inclure des objets spécifiques à certaines industries, comme des outils dans un atelier ou des équipements de sécurité dans une usine.
- ✓ **Utilisation de techniques de transfert d'apprentissage** : Le transfert d'apprentissage permet de réutiliser un modèle pré-entraîné sur un grand ensemble de données et de le fine-tuner sur un ensemble de données spécifique contenant les nouveaux types d'objets. Cela permet de réduire le temps et les ressources nécessaires pour entraîner le modèle.

Intégration avec d'autres systèmes

- ✓ **Systèmes de notification en temps réel** : Intégrer le modèle YOLO avec des systèmes de notification en temps réel, tels que des alertes par SMS, email, ou des applications de messagerie, permettrait de réagir rapidement en cas de détection d'objets importants ou suspects. Cela est particulièrement utile dans les systèmes de surveillance ou de sécurité.
- ✓ **Systèmes de gestion vidéo (VMS)** : L'intégration avec des systèmes de gestion vidéo permettrait de stocker et de gérer efficacement les flux vidéo en conjonction avec les résultats de détection d'objets, facilitant ainsi l'analyse et l'extraction d'informations pertinentes.
- ✓ **Applications IoT (Internet of Things)** : La détection d'objets pourrait être intégrée avec des dispositifs IoT pour des applications comme la gestion des stocks, la surveillance de la chaîne de production, ou les systèmes de maison intelligente, où les objets détectés peuvent déclencher des actions automatisées.

Conclusion

Ce projet a démontré l'efficacité du modèle YOLO pour la détection d'objets en temps réel, atteignant une précision de 88% et un rappel de 85%, avec un temps de traitement moyen de 45 ms par frame, permettant une fréquence de traitement d'environ 22 frames par seconde. Ces résultats montrent que YOLO est capable de fournir des solutions rapides et précises, essentielles pour des applications critiques telles que la surveillance vidéo et la navigation autonome. En réaffirmant l'importance de la détection d'objets, ce projet met en lumière les bénéfices apportés, notamment une meilleure réactivité et une prise de décision plus rapide dans des environnements dynamiques, ce qui est crucial pour la sécurité et l'efficacité opérationnelle. Pour les travaux futurs, plusieurs pistes sont proposées : optimiser davantage les performances du modèle via des techniques de quantification et de pruning pour améliorer la vitesse et l'efficacité, étendre la détection à une variété plus large d'objets en enrichissant l’ensemble de données d'entraînement, améliorer la robustesse du modèle face à des conditions difficiles comme les faibles luminosités et les occlusions, et intégrer le modèle avec des systèmes de notification en temps réel, des systèmes de gestion vidéo, et des dispositifs IoT pour créer des solutions complètes et interconnectées, augmentant ainsi la praticité et l’applicabilité du modèle dans divers domaines.