ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de HONORIS UNITED UNIVERSITIES

HONORIS UNITED UNIVERSITIES

Rapport de projet de fin d'année

4ème année

Ingénierie Informatique et Réseaux

Sous le thème

IDENTIFICATION, SUIVI ET COMPTAGE DES VÉHICULES DANS LEURS DIRECTIONS DE MOUVEMENT RESPECTIVES À L'AIDE D'UNE VIDÉO OU D'UNE CAMÉRA AVEC YOLO ET DEEPSORT EN PYTHON

Réalisé par :
Mohammad Chlouchi, Youssra Ben Ahmadi

Encadré par :
Mme.Ibtissame Aouraghe

Remerciements

En préambule à ce rapport de projet de fin d'année de 4ème, nous tenions à remercier tout particulièrement Madame Aouraghe pour son accueil chaleureux et ses encouragements lors de notre encadrement de ce projet. Merci à vous de nous avoir consacré du temps afin de nous présenter et de nous expliquer le concept de cette application.

Grâce à vous, nous garderons un très bon souvenir de notre projet de fin de quatrième année.

Merci pour tout.

Abstract

Our project is a vehicle identification, tracking, and counting system based on analyzing video or camera feeds. This project showcases the power of computer vision and machine learning techniques. The objective of this project is to provide a robust and accurate solution for traffic management, surveillance systems, and speed detection. The problem we address is the need for real-time and automated vehicle tracking and counting, which is essential for traffic flow analysis, congestion management, and infrastructure planning. Manual methods are time-consuming, prone to errors, and cannot provide real-time insights. Our solution leverages the YOLO object detection algorithm to identify vehicles in the video or camera feed and DeepSORT for robust tracking of these vehicles across frames. By combining these two techniques, we can accurately track vehicles, estimate their trajectories, and count them based on their respective movement directions. This enables authorities to gain valuable insights into traffic patterns, identify congestion hotspots, optimize signal timing, and make data-driven decisions for transportation management.

Résumé

Notre projet est un système d'identification, de suivi et de comptage de véhicules basé sur l'analyse de flux vidéo ou caméra. Ce projet met en valeur la puissance de la vision par ordinateur et des techniques d'apprentissage automatique. L'objectif de ce projet est de fournir une solution robuste et précise pour la gestion du trafic, les systèmes de surveillance et la détection de vitesse. Le problème que nous abordons est le besoin de suivi et de comptage en temps réel et automatisé des véhicules, qui est essentiel pour l'analyse des flux de trafic, la gestion de la congestion et la planification des infrastructures. Les méthodes manuelles prennent du temps, sont sujettes aux erreurs et ne peuvent pas fournir d'informations en temps réel. Notre solution exploite l'algorithme de détection d'objets YOLO pour identifier les véhicules dans le flux vidéo ou caméra et DeepSORT pour un suivi robuste de ces véhicules à travers les images. En combinant ces deux techniques, nous pouvons suivre avec précision les véhicules, estimer leurs trajectoires et les compter en fonction de leurs directions de mouvement respectives. Cela permet aux autorités d'obtenir des informations précieuses sur les modèles de trafic, d'identifier les points chauds de congestion, d'optimiser la synchronisation des signaux et de prendre des décisions basées sur les données pour la gestion des transports.

Introduction

Dans un monde où la mobilité et la logistique jouent un rôle essentiel, il est devenu primordial de disposer d'outils avancés pour identifier, suivre et compter les véhicules en temps réel. Les applications d'identification et de suivi des véhicules permettent aux entreprises de transport, aux agences gouvernementales et à d'autres acteurs de surveiller leurs flottes, d'optimiser les itinéraires, de minimiser les temps d'attente et d'améliorer l'efficacité opérationnelle. La technologie de reconnaissance des véhicules prend également en charge une large gamme de solutions de gestion de la circulation qui permettent de maintenir les véhicules en mouvement, d'améliorer la sécurité routière et d'optimiser la réactivité face aux incidents et infractions liés au trafic routier. Les objectifs principaux de ce projet sont intégration des fonctionnalités avancées d'identification des véhicules, telles que la reconnaissance automatique des plaques d'immatriculation, pour faciliter leur suivi et leur gestion. Aussi la mesure de la distance, la vitesse et la direction d'objets. Pour réaliser le projet de fin d'année nous avons dû passer par certaines étapes avant d'entamer la réalisation : tout d'abord l'étude de l'existant pour avoir une idée générale sur les plateformes existantes sur le marché ainsi que leurs avantages et inconvénients, ensuite une présentation de la méthodologie suivie pour décrire les méthodes et les outils utilisés pour collecter et analyser les données, et pour finir nous allons présenter le résultat avec des interprétations et les limitations de notre travail.

Liste des figures

Figure 1 : Traitement des données sur wialon 10

Figure 2 : Wialon page GPS tracking 11

Figure 3 : Wialon gestion de flotte de véhicules 12

Figure 4 : Wialon géofencing des zones géographiques 12

Figure 5 : Wialon intégration flexibles avec d'autres systèmes 13

Figure 6 : Hikvision page d'accueil des produits 15

Figure 7 : Les technologies adoptées par Hikvision 15

Figure 8 : Résultat d'exécution du code de notre application 31

Table des matières

Chapitre I : Contexte général **8**

1.1) Introduction 9

1.2) Etude de l'existant 9

1.3) Problématique 17

1.4) Solution proposée 17

1.5) Conclusion 18

Chapitre II : Méthodologie adoptée **19**

2.1) Introduction 20

2.2) Système de travail suivi 20

2.3) Bibliothèques utilisées 24

2.4) Conclusion 32

Chapitre III : Réalisation **33**

3.1) Introduction 34

3.2) Résultats 34

3.3) Interprétations 35

3.4) Limitations de travail 41

3.5) Conclusion 41

Chapitre I : Contexte général

1.1) Introduction :

Avant de commencer le travail sur notre projet, nous avons dû passer par plusieurs étapes qui seront détaillées dans ce rapport. La première étape, représentée par ce chapitre, a été d'analyser les applications de suivi de véhicules existantes, principalement Wialon et Hikvision, afin de découvrir leurs avantages et leurs inconvénients, pour qu'on puisse finalement créer un système qui comble les lacunes du marché en résolvant potentiellement toutes leurs faiblesses.

1.2) Etude de l'existant :

A. Wialon :

Une plateforme de suivi GPS et de gestion de flotte développée par la société Gurtam. C'est l'une des principales applications de Gurtam, largement utilisée dans le domaine de la gestion de flotte et de la géolocalisation. Wialon offre une variété de fonctionnalités pour l'identification, le suivi et le comptage des véhicules en temps réel. [1]

- Le traitement des données sur Wialon [2] :

Les données sont un message qui est stocké et transmis par un appareil installé sur une unité. Ces messages comprennent :

- Le temps ;
- Les informations provenant du module GPS : les coordonnées, la vitesse, l'altitude, le nombre de satellites ;
- Les paramètres des capteurs externes et internes : le niveau de carburant du capteur de niveau de carburant, le nombre de tours de roue de l'odomètre, etc.
- Wialon reçoit ces données, les modifie si nécessaire et les traite.

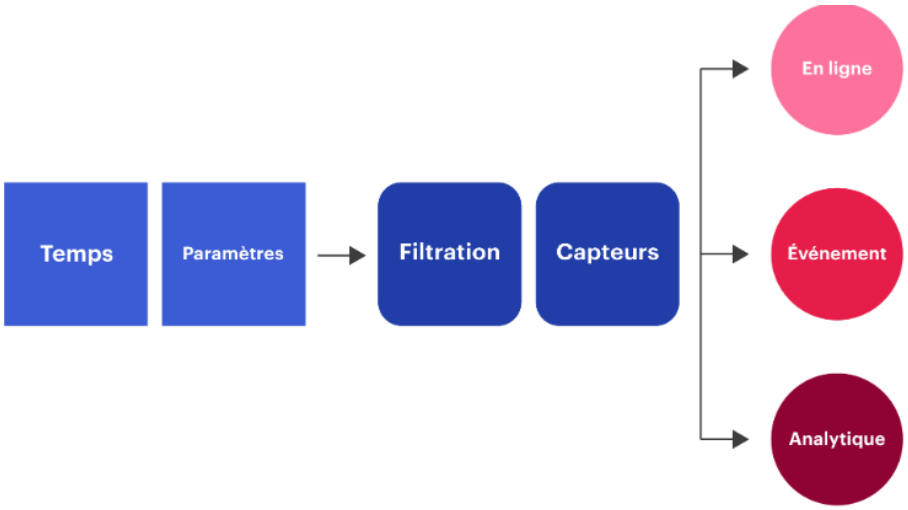

Figure 1 : Traitement des données sur wialon

- Temps dans Wialon :

Pour Wialon, le temps est extrêmement important, puisque Wialon utilise le temps pour trier les messages.

Comment cela fonctionne : le tracker génère un message en utilisant le format de temps Unix et l'envoie au serveur.

- Configuration de l'appareil.
- Synchronisation d'horloge

- Paramètres du capteur :

Les paramètres sont des valeurs dans les messages qui décrivent l'état de l'unité surveillée et des appareils qui y sont installés.

Les capteurs déterminent la logique du traitement des paramètres. Chacun des types de capteurs à des algorithmes qui lui sont associés dans Wialon. Par exemple, les rapports pour la température et le poids ont des colonnes séparées que Wialon remplit automatiquement en fonction du type de capteur.

Comment les capteurs traitent les données ?

-Affichage des données avec des formules mathématiques : Dans le champ où le paramètre est spécifié, l'utilisateur peut indiquer une formule.
-Calcul de tableaux : Il permet aux utilisateurs d'afficher des dépendances simplifiées en utilisant seulement quelques-uns des points que l'intégrateur détermine en pratique.
-Validation : Grâce à la validation, l'utilisateur peut relier les valeurs de plusieurs capteurs entre elles. Pour cela, on utilise non seulement des opérations mathématiques mais aussi des opérations logiques.

Trois principaux types de traitement des données

Il existe trois grandes méthodes de traitement des données dans Wialon auxquelles les utilisateurs ont le plus souvent recours. Deux d'entre elles - en ligne et analytique - existent depuis un certain temps. La troisième méthode - les événements - est apparue récemment.

Ces méthodes utilisent les mêmes messages.

- En ligne :

- Onglet Surveillance. L'onglet présente une liste d'unités avec un grand nombre d'indicateurs correspondants. L'utilisateur clique sur l'icône de l'unité et obtient des données étendues, par exemple les valeurs du capteur de niveau de carburant. Lorsque les valeurs changent, Wialon met à jour les données.

- Dashboard. Les informations telles que les géozones et les unités qui s'y trouvent ou l'état du mouvement se présentent sous la forme de diagrammes avec des données en ligne.
- Notifications. La méthode permet aux utilisateurs finaux de calculer la durée de l'état de l'unité.

- Analytique : messages, traces et rapports

- L'analyse peut être opposée à la méthode en ligne d'une certaine manière : la méthode en ligne traite chaque message entrant ici et maintenant, tandis que l'analyse travaille avec une période complète.

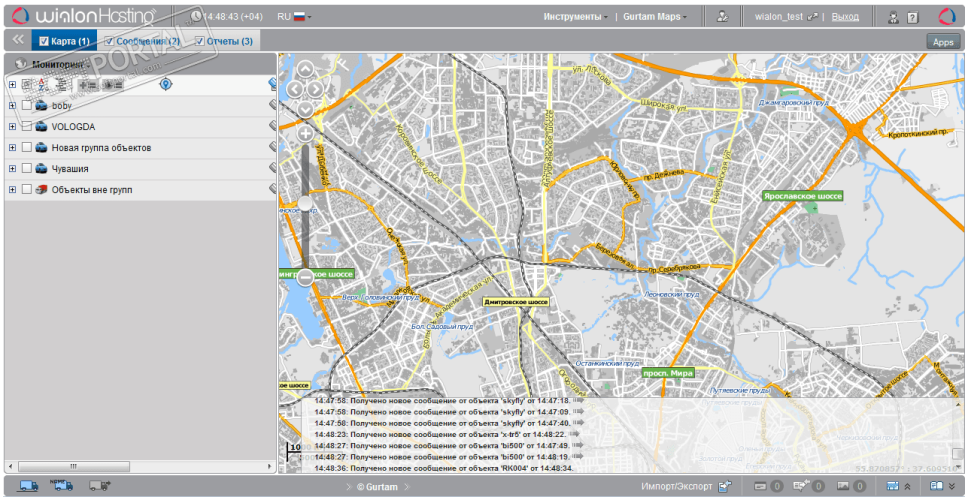

Figure 2 : Wialon page GPS tracking

a. Les avantages :

- Suivi GPS en temps réel : Wialon permet de suivre la localisation en temps réel des véhicules équipés de dispositifs GPS. Vous pouvez voir la position des véhicules sur une carte, obtenir des mises à jour régulières sur leur emplacement et surveiller leur itinéraire.
- Gestion de flotte : L'application permet de gérer efficacement une flotte de véhicules en fournissant des informations détaillées sur les performances, l'utilisation du carburant, les temps de conduite, etc. Vous pouvez également définir des alertes personnalisées pour surveiller les comportements de conduite et les événements spécifiques.

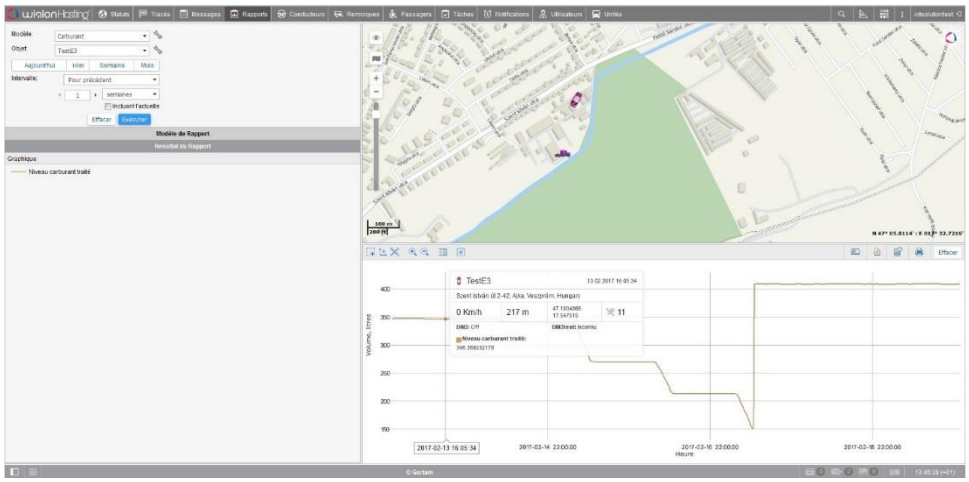

Figure 3 : Wialon gestion de flotte de véhicules

- Géofencing : La fonction de géofencing permet de définir des zones géographiques spécifiques et de recevoir des alertes lorsque les véhicules entrent ou sortent de ces zones. Cela peut être utile pour le suivi des livraisons, la gestion des itinéraires ou la surveillance des véhicules dans des zones restreintes.

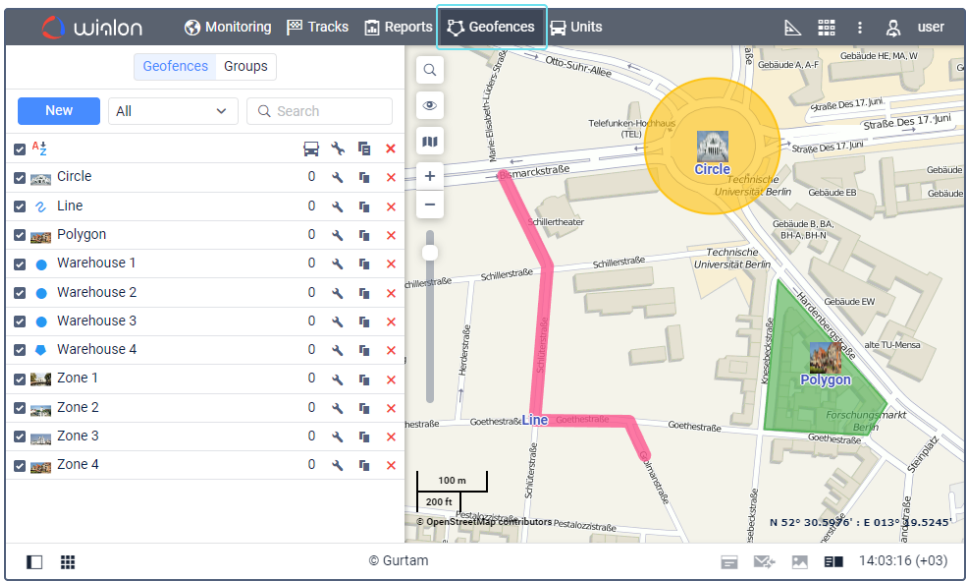

Figure 4 : Wialon géofencing des zones géographiques

- Intégration avec d'autres systèmes : Wialon offre des options d'intégration flexibles avec d'autres systèmes, tels que les CRM, les systèmes de gestion de carburant, les systèmes de navigation, etc. Cela permet une gestion efficace des données et une collaboration transparente avec d'autres solutions.

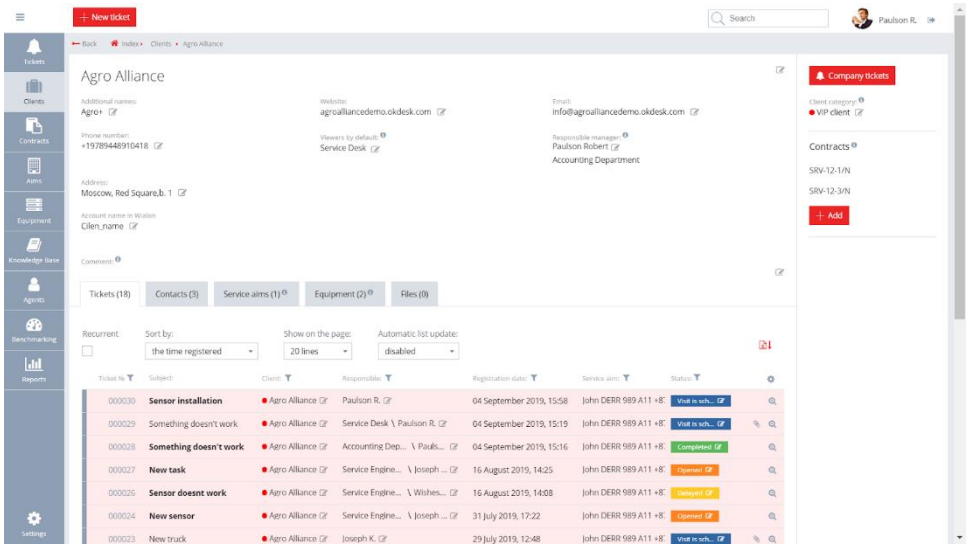

Figure 5 : Wialon intégration flexibles avec d'autres systèmes

Les détails spécifiques sur les données et les modèles de machine learning utilisés dans Wialon ne sont pas explicitement divulgués dans les informations disponibles publiquement. Cependant, on propose une idée générale des types de données et des approches de machine learning qui pourraient être utilisées dans un système de suivi de flotte et de gestion de la géolocalisation comme Wialon :

1. Données de suivi des véhicules.

2. Données historiques de flotte.

3. Modèles prédictifs.

4. Analyse des comportements de conduite.

b. Les inconvénients :

- Données illisibles : Les informations que Wialon reçoit ne sont pas toujours claires pour l'utilisateur. Lorsque l'utilisateur voit la valeur temperature13, il comprend : on parle de température. Mais la température : du moteur, du carburant ou de l'air ? Celsius ou Fahrenheit ?

- Wialon tronque un horodatage à la seconde mais n'indique pas les millisecondes : En même temps, les trackers populaires comme Teltonika, Ruptela, Galileosky peuvent générer plus d'un message par seconde. Avec un flux important de messages, cela peut entraîner une certaine confusion, car Wialon ne dispose d'aucun autre critère pour trier les messages, à l'exception du temps.

B. Hikvision :

L'application Hikvision LAPI enregistre la plaque d'immatriculation d'un véhicule en temps réel et la compare ou l'ajoute à une liste prédéfinie. Une fois qu'une plaque d'immatriculation a été reconnue et enregistrée, la mesure appropriée est effectuée (par exemple, ouverture d'une porte, ajout d'un coût ou déclenchement d'une alerte). La reconnaissance des véhicules de Hikvision utilise un algorithme d'apprentissage approfondi pour distinguer les véhicules de petite taille et de grande taille, avec également la possibilité de détecter la marque, le modèle et la couleur. Le système peut être programmé pour rechercher des anomalies telles que des voitures circulant dans des voies réservées aux bus ou des véhicules sans plaque d'immatriculation. [3]

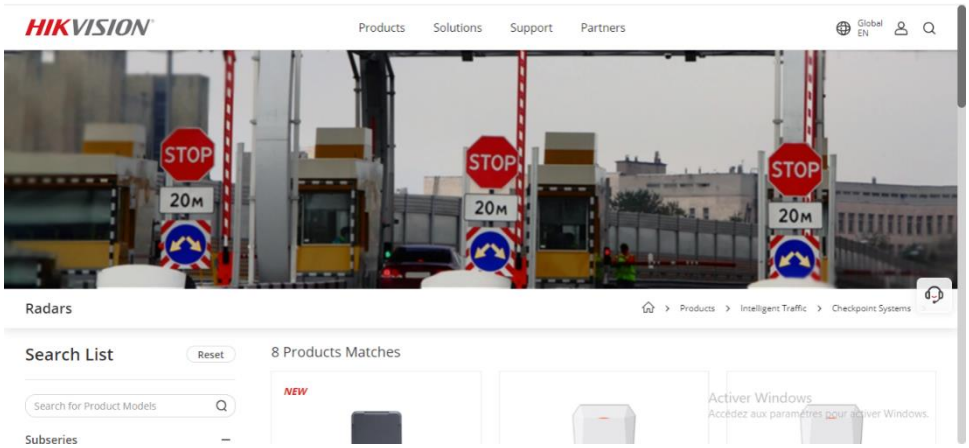

Figure 6 : Hikvision page d'accueil des produits

- **Le traitement des données Hikvision**

Hikvision s'impose comme un véritable pionnier dans le domaine de la sécurité et tout ce qui s'y rapporte. Avec des caméras intelligentes et des solutions d'infrastructure leaders sur le marché, développées avec l'intelligence artificielle, intègrent un large éventail d'outils conçus pour faciliter des prises de décisions plus avisées, du comptage des personnes à la reconnaissance faciale, en passant par la reconnaissance des véhicules et les alertes de sécurité automatisées. Les technologies intelligentes de Hikvision possèdent des applications quasiment illimitées. [4]

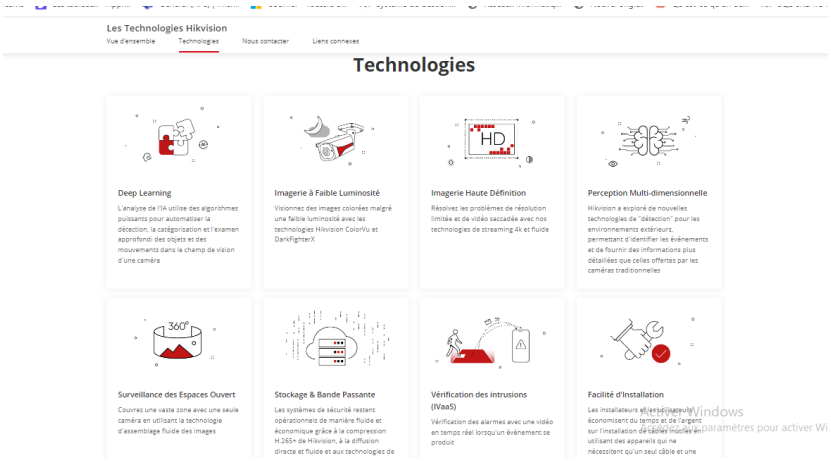

Figure 7 : Les technologies adoptées par Hikvision

a. Les avantages :

- Hikvision a adopté une approche globale de la cybersécurité. Il assure une cybersécurité maximale sur l'ensemble de leur chaîne de valeur et sur l'ensemble du cycle de vie de leurs produits. Cette approche s'appelle : Secure-by-Design et est une priorité absolue chez Hikvision. [5]

- Les caméras IP Hikvision offrent des images qualitatives dans toutes les conditions d'éclairage, tout en minimisant les besoins en stockage et bande passante. La polyvalence de la gamme de caméras Hikvision en fait le choix idéal pour une multitude de scénarios d'application.

- Développement durable Hikvision a toujours pour objectif de responsabiliser les clients et de créer de la valeur pour la société grâce à l'innovation technologique.

b. Les inconvénients :

- Des mauvaises mises à jour de l'application [6]

1.3) Problématique :

Le problème que nous abordons est le besoin de suivi et de comptage en temps réel et automatisé des véhicules, qui est essentiel pour l'analyse des flux de trafic, la gestion de la congestion et la planification des infrastructures. Les méthodes manuelles prennent du temps, sont sujettes aux erreurs et ne permettent pas d'obtenir de l'information en temps réel.

1.4) Solution proposée :

On propose un algorithme de détection d'objets YOLO pour identifier les véhicules dans le flux vidéo ou caméra et DeepSORT pour un suivi robuste de ces véhicules à travers les images. Ces

deux techniques, nous permettent de suivre avec précision les véhicules, estimer leurs trajectoires et les compter en fonction de leurs directions de mouvement respectifs. Cela permet aux autorités d'obtenir des informations précieuses sur les modèles de trafic, d'identifier les points chauds de congestion, d'optimiser la synchronisation des signaux et de prendre des décisions basées sur les données pour la gestion des transports.

1.5) Conclusion:

Lors de cette première partie, nous avons cité les différents avantages des plateformes existantes sur le marché ainsi que leurs points faibles, et à partir de ces derniers nous avons issu la problématique générale et par la suite nous avons proposé notre propre vision afin de résoudre ce problème que nous allons voir dans les chapitres suivants.

Chapitre II : Méthodologie adoptée

2.1) Introduction :

La deuxième étape lors de la réalisation de notre projet, listée dans ce chapitre, était l'établissement des différentes étapes de travail et l'identification des différentes bibliothèques dont nous aurons besoin afin de réaliser notre projet passant par cv2, torch, numpy et tracker. Ces quatre bibliothèques mettent en valeur les différentes fonctionnalités de notre application ainsi que des explications simples et détaillées.

2.2) Système de travail suivi :

Lors de la réalisation de notre projet, nous avons suivi plusieurs étapes pour extraire une image à partir d'une vidéo et calculer la vitesse des véhicules détectés. Tout d'abord, nous avons utilisé la bibliothèque OpenCV pour lire chaque image de la vidéo et les traiter. Ensuite, nous avons utilisé le modèle YOLOv5, un modèle de détection d'objets pré-entraîné, pour identifier les véhicules dans chaque image. Les résultats de détection ont été extraits et stockés dans une liste. Pour faire le suivi en temps réel des véhicules, nous avons mis en place une classe `Tracker` qui utilise un algorithme de suivi des objets pour mettre à jour les coordonnées des véhicules détectés à chaque image. Les véhicules détectés ont été suivis en utilisant des polygones définissant des zones de détection, et leur entrée et leur sortie dans ces zones ont été enregistrées dans des dictionnaires avec les temps correspondants. En utilisant les informations de détection et de suivi, nous avons calculé la vitesse des véhicules en mesurant la distance parcourue et le temps passé dans chaque zone de détection. Le code a également utilisé les bibliothèques Python YOLOv5 et DeepSort pour la détection d'objets et le suivi des véhicules respectivement. Le modèle YOLOv5 est basé sur l'apprentissage automatique (deep learning) et a été pré-entraîné sur un ensemble de données contenant différentes classes d'objets, y compris les véhicules. Le suivi des objets avec DeepSort repose également sur des techniques d'apprentissage automatique pour suivre les véhicules en fonction de leurs caractéristiques et de leurs mouvements.

- **Identification et importation des bibliothèques nécessaires :**

  - Les bibliothèques utilisées comprennent `cv2` (OpenCV), `torch`, `numpy`, `tracker` et `time`.

  - `cv2` est utilisé pour la capture vidéo, le traitement d'image et l'affichage.

  - `torch` est utilisé pour charger le modèle YOLOv5 pré-entraîné.

  - `numpy` est utilisé pour effectuer des opérations sur les tableaux d'images.

  - `tracker` est utilisé pour le suivi des objets détectés.

  - `time` est utilisé pour mesurer les temps de présence des véhicules.

- **Chargement du modèle YOLOv5 :**

  - Le modèle YOLOv5 pré-entraîné est chargé à l'aide de la fonction `torch.hub.load` et stocké dans la variable `model`.

  - Le modèle est chargé avec l'architecture "yolov5s" et les poids pré-entraînés.

- **Ouverture du flux vidéo :**

  - La vidéo est ouverte à l'aide de `cv2.VideoCapture` et stockée dans une variable.

  - La vidéo est lue image par image dans une boucle.

- **Initialisation des variables et structures de données :**

  - Des variables sont initialisées pour compter le nombre total de frames traitées, ainsi que pour stocker les résultats de suivi des véhicules.

- Une liste de noms de classes d'objets est définie, contenant les noms des classes pour les véhicules à détecter.
- Des dictionnaires sont créés pour stocker les informations sur les véhicules détectés dans les différentes zones d'intérêt.
- Des variables sont définies pour enregistrer les temps de présence des véhicules dans les zones d'intérêt et leurs vitesses.
- Une variable est utilisée pour stocker le nom de la classe d'objet actuellement détectée.

- **Définition de la fonction de rappel pour la gestion des événements de la souris :**

- Une fonction de rappel nommée `POINTS` est définie pour capturer les coordonnées des points cliqués par la souris dans la fenêtre d'affichage de la vidéo.
- Cette fonction est associée à la fenêtre en utilisant la fonction `cv2.setMouseCallback`.

- **Définition des zones d'intérêt :**

- Deux zones d'intérêt sont définies en utilisant des coordonnées de points qui forment les contours des zones.
- La distance entre les points des zones est calculée à l'aide de la formule de distance euclidienne et stockée dans les variables.
- Des ensembles sont utilisés pour stocker les identifiants des véhicules détectés dans chaque zone.

- **Boucle principale de traitement du flux vidéo :**

- Une boucle infinie est exécutée pour traiter chaque frame de la vidéo.

- La fonction `cap.read()` est utilisée pour lire une frame du flux vidéo, et la boucle se termine lorsque la fin de la vidéo est atteinte.
- La variable `count` est utilisée pour sauter certaines frames afin de réduire le nombre d'images traitées et d'améliorer les performances.
- La frame est redimensionnée à une taille de 1020x600 pixels à l'aide de `cv2.resize`.

- **Détection et suivi des véhicules :**
  - Le modèle YOLOv5 est appliqué à la frame en utilisant `results = model(frame)`.
  - Les résultats de détection sont convertis en un DataFrame pandas et parcourus pour extraire les coordonnées des boîtes englobantes des véhicules.
  - Les informations de détection sont stockées dans une liste pour le suivi ultérieur des véhicules.
  - Le suivi des véhicules est effectué en utilisant le tracker `tracker.update(liste)`, qui retourne les boîtes englobantes mises à jour.

- **Affichage des véhicules détectés et calcul de la vitesse :**
  - Les boîtes englobantes mises à jour sont utilisées pour afficher les véhicules détectés dans la frame à l'aide de `cv2.rectangle` et `cv2.putText`.
  - Si le nom de la classe d'objet est présent dans `classNames`, il est affiché avec l'identifiant du véhicule détecté.
  - Des cercles sont dessinés autour des points de suivi des véhicules à l'aide de `cv2.circle`.
  - Les véhicules qui entrent et sortent des zones d'intérêt sont identifiés en utilisant `cv2.pointPolygonTest`.

- Les temps de présence des véhicules dans les zones d'intérêt sont calculés en utilisant `time.time()` et en les soustrayant des moments d'entrée/sortie précédents.
- Les vitesses des véhicules sont calculées en utilisant la distance entre les points des zones et le temps de présence, et converties en mètres par seconde et en kilomètres par heure.
- Les vitesses des véhicules sont affichées à l'aide de `cv2.putText`.

- **Affichage des zones d'intérêt et des statistiques :**

- Les contours des zones d'intérêt sont affichés à l'aide de `cv2.polylines`.
- Le nombre de véhicules détectés dans chaque zone d'intérêt est calculé en utilisant la longueur des ensembles `zone_1` et `zone_2`, et affiché à l'aide de `cv2.putText`.
- Les statistiques totales, y compris le nombre total de véhicules détectés dans les deux zones, sont affichées à l'aide de `cv2.putText`.

- **Affichage de la frame et gestion des événements clavier :**

- La frame traitée est affichée dans une fenêtre intitulée "FRAME" à l'aide de `cv2.imshow`

- La boucle se poursuit jusqu'à ce qu'une touche soit enfoncée. Si la touche "Esc" est enfoncée (ayant la valeur 27), la boucle se termine et la vidéo est libérée à l'aide de `cap.release()` et toutes les fenêtres sont fermées avec `cv2.destroyAllWindows()`.

2.3) Bibliothèques utilisées :

2.3.1) cv2 :

cv2 est une bibliothèque open-source populaire utilisée pour la vision par ordinateur et le traitement d'images en Python. Voici quelques détails sur cv2:

Fonctionnalités de base :

- cv2 fournit une large gamme de fonctions pour lire, écrire et manipuler des images et des vidéos.
- Il permet de charger des images à partir de fichiers dans différents formats tels que JPEG, PNG, BMP, etc., et de les afficher à l'écran.
- Il peut être utilisé pour capturer des flux vidéo à partir de caméras en direct, des vidéos préenregistrées ou même des fichiers vidéo.
- cv2 fournit des fonctionnalités pour effectuer des opérations de base sur les images telles que le redimensionnement, la rotation, le recadrage, la conversion de couleurs, etc.
- Il offre des fonctionnalités de dessin pour ajouter des formes géométriques, des textes et d'autres annotations aux images.

Traitement d'images :

- cv2 propose une large gamme d'opérations de traitement d'images, y compris le flou, la mise au point, l'égalisation d'histogramme, la normalisation, la correction de contraste, etc.
- Il permet de réaliser des opérations arithmétiques et logiques sur les images, comme l'addition, la soustraction, la multiplication, la fusion, etc.
- Il fournit des outils pour détecter les contours, les coins, les lignes droites et autres caractéristiques dans une image.
- cv2 inclut également des fonctionnalités pour la détection de visages, la détection de formes, la segmentation d'images, etc.
- -Vision par ordinateur :
- cv2 offre des fonctionnalités pour la détection et le suivi d'objets dans des images ou des flux vidéo en utilisant des algorithmes tels que Haar cascades, YOLO, etc.
- Il peut être utilisé pour effectuer la correspondance de points entre des images, la calibration de caméra, la stéréovision, la reconstruction 3D, etc.

- cv2 prend en charge l'apprentissage automatique avec des fonctionnalités pour la création, l'entraînement et l'utilisation de modèles de reconnaissance d'objets, de classification d'images, etc.
- Il permet également de travailler avec des flux vidéo en temps réel, en traitant chaque frame individuellement et en affichant les résultats en direct.

cv2 est une bibliothèque puissante et largement utilisée dans la communauté de la vision par ordinateur. Elle offre une grande variété de fonctionnalités pour le traitement et l'analyse d'images, le suivi d'objets, la détection de formes et bien d'autres tâches liées à la vision par ordinateur. [7]

2.3.2) numpy :

Numeric Python «numpy» est une bibliothèque open-source populaire pour le calcul numérique en Python. Elle fournit des structures de données et des fonctions pour manipuler efficacement des tableaux multidimensionnels (appelés ndarray) et effectuer des opérations mathématiques sur ces tableaux.

Voici quelques caractéristiques et fonctionnalités clés de NumPy :

- Tableaux multidimensionnels : NumPy introduit un nouveau type d'objet appelé ndarray, qui est un tableau multidimensionnel homogène. Les tableaux ndarray sont plus efficaces en termes de stockage et de performances que les listes Python standard pour les opérations de calcul numérique.

- Opérations mathématiques : NumPy offre une large gamme de fonctions mathématiques pour effectuer des opérations telles que l'addition, la soustraction, la multiplication, la division, l'exponentiation, les fonctions trigonométriques, les fonctions logarithmiques, etc. sur des tableaux. Ces opérations sont vectorisées, ce qui signifie qu'elles sont appliquées élément par élément sans nécessiter de boucles explicites.

- Broadcasting : NumPy permet de réaliser des opérations entre des tableaux de formes différentes de manière transparente en utilisant la fonctionnalité de broadcasting. Le broadcasting étend automatiquement les tableaux de formes différentes pour qu'ils aient des dimensions compatibles avant d'effectuer l'opération.

- Indexation et slicing : NumPy permet d'accéder aux éléments individuels d'un tableau à l'aide d'index et de réaliser des opérations de découpage (slicing) pour extraire des sous-ensembles de tableaux.

- Algèbre linéaire : NumPy fournit des fonctions pour effectuer des opérations d'algèbre linéaire telles que les calculs de vecteurs et de matrices, les décompositions (LU, QR, valeur singulière), les inverses, les résolutions de systèmes linéaires…

- Génération de nombres aléatoires : NumPy propose des outils pour générer des nombres aléatoires selon différentes distributions, ce qui est utile pour les simulations et les tâches liées à l'apprentissage automatique.

- Intégration avec d'autres bibliothèques : NumPy est la base de nombreuses autres bibliothèques scientifiques en Python, telles que Pandas, SciPy et scikit-learn. Il offre une interface commune pour le traitement des données numériques, facilitant ainsi l'intégration entre ces bibliothèques.

En résumé, NumPy est une bibliothèque essentielle pour les calculs numériques en Python. Elle fournit des structures de données efficaces pour les tableaux multidimensionnels ainsi que des fonctions mathématiques et des outils pour effectuer des opérations numériques avancées de manière rapide et efficace. [8]

2.3.3) torch :

torch ou PyTorch est une bibliothèque open-source d'apprentissage automatique (machine learning) basée sur le langage de programmation Python. Elle est principalement utilisée pour développer des modèles d'apprentissage profond (deep learning). PyTorch est largement utilisé

et apprécié par la communauté de recherche en apprentissage automatique pour sa flexibilité, sa facilité d'utilisation et ses performances élevées.

Voici quelques caractéristiques et fonctionnalités clés de PyTorch :

- Tenseurs : PyTorch utilise une structure de données appelée tenseur pour représenter et manipuler les données. Les tenseurs sont similaires aux tableaux multidimensionnels de NumPy et fournissent une interface intuitive pour effectuer des calculs numériques.

- Calcul automatique des gradients : PyTorch prend en charge le calcul automatique des gradients (ou différentiation automatique) à l'aide de la technique de la différenciation automatique basée sur le graphe de calcul. Cela permet de calculer efficacement les gradients des fonctions et des modèles, ce qui est essentiel pour l'optimisation des paramètres lors de l'entraînement des réseaux de neurones.

- Construction dynamique des graphes de calcul : PyTorch utilise un modèle de construction dynamique des graphes de calcul. Cela signifie que le graphe de calcul est construit à la volée lors de l'exécution du code, ce qui offre une grande flexibilité pour la conception et la modification des modèles.

- Intégration avec les GPU : PyTorch prend en charge l'utilisation des unités de traitement graphique (GPU) pour accélérer les calculs. Il fournit une interface simple pour transférer les tenseurs sur un GPU et effectuer des calculs parallèles, ce qui permet d'exploiter efficacement la puissance de calcul des GPU.

- Modules et optimiseurs : PyTorch fournit une grande variété de modules pré-construits pour la création de réseaux de neurones, tels que les couches linéaires, les fonctions d'activation, les fonctions de perte, etc. Il propose également des optimiseurs pour la mise à jour des paramètres des modèles lors de l'apprentissage, tels que Stochastic Gradient Descent (SGD), Adam, etc.

- Intégration avec d'autres bibliothèques : PyTorch s'intègre facilement avec d'autres bibliothèques populaires d'apprentissage automatique telles que NumPy et SciPy. Cela permet d'importer et de manipuler des données provenant de différentes sources et de bénéficier des fonctionnalités complémentaires offertes par ces bibliothèques.
- Communauté active : PyTorch bénéficie d'une communauté de développeurs active et d'un écosystème en pleine expansion. De nombreux chercheurs et praticiens utilisent PyTorch pour leurs projets d'apprentissage automatique, ce qui se traduit par une documentation détaillée, des tutoriels, des exemples de code et une assistance en ligne.

En résumé, PyTorch est une bibliothèque d'apprentissage automatique populaire et puissante, axée sur le deep learning. Elle offre des fonctionnalités avancées pour la manipulation de tenseurs, le calcul automatique des gradients, la construction dynamique des graphes. [9]

2.3.4) DeepSORT :

DeepSORT est une extension de l'algorithme SORT (Simple Online and Realtime Tracking) qui utilise des techniques de deep learning pour améliorer la précision et la robustesse du suivi d'objets, il combine la détection d'objets (par exemple, avec YOLO) et le suivi des objets pour obtenir des identifiants cohérents pour chaque objet dans une séquence d'images ou de vidéos.

Cette bibliothèque utilise un réseau de neurones pour extraire des caractéristiques (features) des objets détectés, puis applique un algorithme de suivi basé sur la distance pour associer les objets d'un frame à l'autre.

DeepSORT est capable de gérer des scénarios complexes avec des objets qui entrent/sortent du champ de vision, se chevauchent, changent de forme, etc.
Il est largement utilisé dans des applications telles que la surveillance vidéo, la reconnaissance de gestes, la conduite autonome, etc. [10]

2.3.5) YOLO :

YOLO (You Only Look Once) est une famille d'architectures de réseaux de neurones convolutifs utilisée pour la détection d'objets en temps réel. YOLO5 fait référence à la version 5 de YOLO, qui est la dernière itération de cette famille.

YOLO5 repose sur le principe de la détection d'objets en une seule passe, ce qui signifie qu'il analyse l'image complète et prédit les emplacements et les classes des objets détectés simultanément, plutôt que de diviser l'image en régions et de les classifier séparément. Cela permet à YOLO5 d'être rapide et efficace pour la détection d'objets en temps réel.

Les principales caractéristiques de YOLO5 sont les suivantes :

- Architecture : YOLO5 utilise une architecture de réseau de neurones convolutifs basée sur des couches de convolution, de mise en commun (pooling) et de mise en commun adaptative (adaptive pooling). Elle est conçue pour extraire des caractéristiques pertinentes à différentes échelles spatiales dans l'image.

- Backbone : YOLO5 utilise un réseau de neurones pré-entraîné, généralement basé sur des architectures comme Darknet ou CSPDarknet, en tant que backbone pour extraire les caractéristiques de l'image en entrée. Le backbone peut être ajusté ou remplacé en fonction des besoins spécifiques.

- Extraction de caractéristiques multi-échelles : YOLO5 utilise des couches de mise en commun adaptative pour agréger les caractéristiques à différentes échelles spatiales. Cela permet de détecter des objets de différentes tailles dans l'image.

- Prédiction des boîtes englobantes : YOLO5 prédit les boîtes englobantes (bounding boxes) des objets détectés en termes de coordonnées (x, y, largeur, hauteur) et d'une probabilité d'objet présent dans chaque boîte.

- Classification des objets : YOLO5 attribue également une classe prédite à chaque boîte englobante, indiquant le type d'objet détecté. Le nombre de classes dépend du jeu de données sur lequel le modèle a été entraîné.

- Entraînement et optimisation : YOLO5 est généralement entraîné en utilisant des techniques d'apprentissage supervisé et d'optimisation, telles que la rétropropagation du gradient et l'optimisation par descente de gradient stochastique (SGD). Les modèles pré-entraînés sur de grands ensembles de données, comme COCO ou ImageNet, peuvent être utilisés comme point de départ pour un entraînement plus spécifique.

YOLO5 est apprécié pour sa rapidité et sa précision en détection d'objets, et il est couramment utilisé dans des applications telles que la surveillance vidéo, la détection de piétons, la conduite autonome et la vision par ordinateur en temps réel. [11]

2.3.6) math :

La bibliothèque mathématique en Python est une collection d'outils et de fonctions qui permettent d'effectuer des opérations mathématiques courantes [12] . Une fois la bibliothèque importée, vous pouvez utiliser ses fonctions pour effectuer diverses opérations mathématiques.

Voici quelques exemples des fonctions les plus couramment utilisées :

- math.sqrt(x) : Cette fonction renvoie la racine carrée de `x`.

- math.exp(x) : Cette fonction renvoie `e` à la puissance `x`, où `e` est la base du logarithme naturel.

- math.log(x) : Cette fonction renvoie le logarithme naturel (base `e`) de `x`.

2.3.7) time :

La bibliothèque "time" en Python est une bibliothèque intégrée qui fournit des fonctionnalités pour travailler avec le temps. Elle permet de mesurer le temps écoulé, de formater et de manipuler les dates et les heures. Une fois la bibliothèque importée, vous pouvez utiliser ses fonctions pour effectuer différentes opérations liées au temps. [13]

Voici quelques exemples des fonctions les plus couramment utilisées :

- time.time() : Cette fonction renvoie le temps écoulé en secondes depuis le 1er janvier 1970 (également appelé "temps Unix"). Elle est souvent utilisée pour mesurer la durée d'exécution d'un programme.

- time.sleep(seconde) : Cette fonction suspend l'exécution du programme pendant le nombre de secondes spécifié. Elle est utile lorsque vous avez besoin de faire une pause dans votre programme.

- time.localtime() : Cette fonction renvoie l'heure locale sous forme d'un objet de structure de temps contenant les informations sur l'année, le mois, le jour, l'heure, les minutes, les secondes, le jour de la semaine, le jour de l'année...

2.4) Conclusion :

Dans ce chapitre, nous avons proposé trois variantes des schémas de conception en utilisant la technologie UML qui ont mis en valeur et clarifié les différentes interactions qui prennent places entre le côté client/visiteur et côté système. Et afin de donner naissance à ces différentes fonctionnalités, nous avons utilisés quelques technologies, chacune pour une partie spécifique du projet, que nous allons introduire dans le chapitre suivant.

Chapitre III : Réalisation

3.1) Introduction :

La troisième étape dans notre projet, expliquée dans ce dernier chapitre, est la partie réalisation. Dans cette section du rapport, nous allons introduire les différentes technologies qui nous étaient utiles lors de la création de notre application, ainsi que des exemples des différentes fonctionnalités que nous proposons dans notre application.

3.2) Résultats :

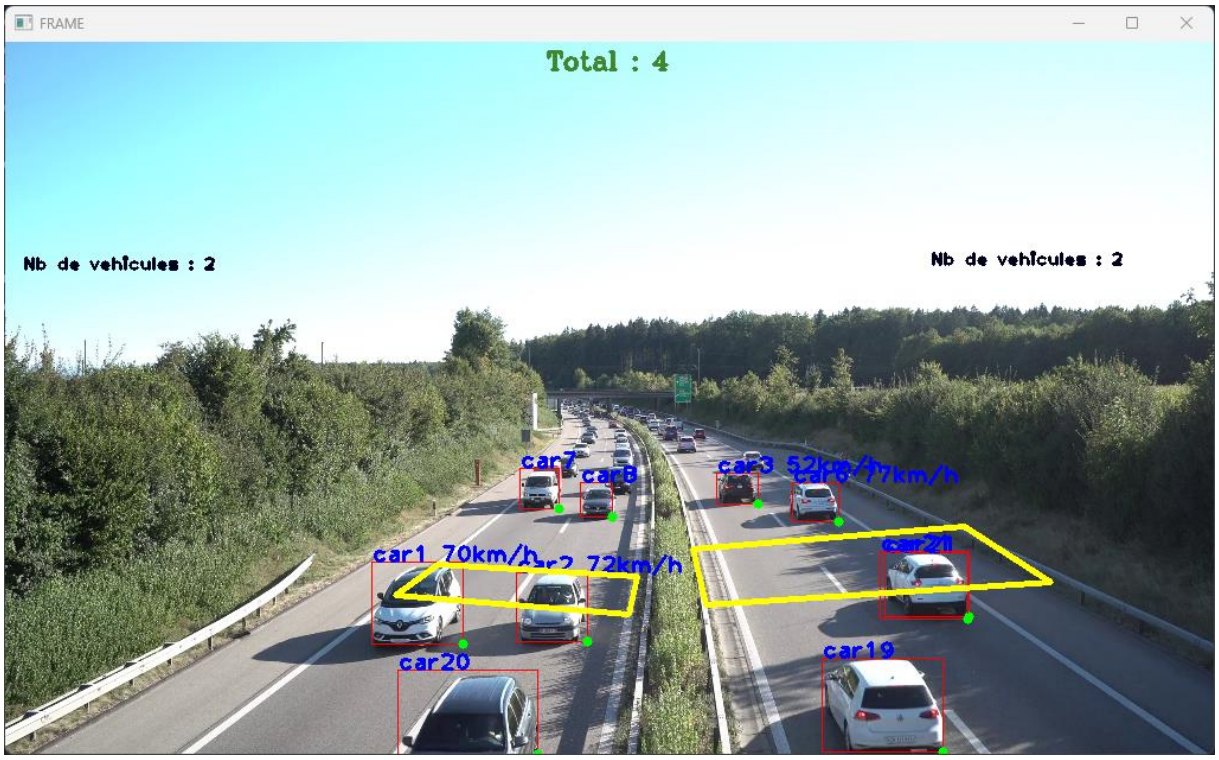

Figure 8 : Résultat d'exécution du code de notre application

3.3) Interprétations:

Ce code représente une classe appelée "Tracker" qui est utilisée pour suivre des objets détectés dans une scène. Voici une interprétation détaillée du code :

1. Tout d'abord, le module math est importé pour utiliser la fonction hypot(), qui calcule l'hypoténuse d'un triangle à partir de ses côtés (utilisé pour calculer la distance entre deux points).

2. Le constructeur de la classe "Tracker" est défini avec la méthode spéciale __init__(). Cette méthode est appelée lorsqu'une instance de la classe est créée. À l'intérieur de cette méthode :

- Un dictionnaire vide appelé "center_points" est créé pour stocker les positions centrales des objets détectés.
- Une variable appelée "id_count" est initialisée à 0 pour compter les IDs des objets détectés.

3. La méthode "update" est définie. Cette méthode prend en paramètre une liste d'objets_rect, qui représente les coordonnées des boîtes englobantes (rectangles) des objets détectés dans la scène. À l'intérieur de cette méthode :

- Une liste vide appelée "objects_bbs_ids" est créée pour stocker les boîtes englobantes des objets ainsi que leurs IDs.
- Pour chaque rectangle dans la liste des objets_rect :
- Les coordonnées x, y, largeur (w) et hauteur (h) du rectangle sont extraites.
- Les coordonnées du centre du rectangle sont calculées en utilisant les formules cx = (x + x + w) // 2 et cy = (y + y + h) // 2. Cela calcule les coordonnées du point au milieu du rectangle.
- Ensuite, une boucle est utilisée pour parcourir chaque ID et position de centre dans le dictionnaire "center_points".

- La distance entre les coordonnées du centre calculé et les coordonnées du centre stockées dans le dictionnaire est calculée à l'aide de la fonction math.hypot().
- Si la distance est inférieure à 35 (ce qui signifie que l'objet actuel est suffisamment proche d'un objet déjà détecté), les coordonnées du centre dans le dictionnaire sont mises à jour avec les nouvelles coordonnées calculées.
- La boîte englobante de l'objet ainsi que son ID sont ajoutés à la liste "objects_bbs_ids".
- La variable "same_object_detected" est définie sur True pour indiquer qu'un objet similaire a été détecté.
- La boucle est interrompue car l'objet a été trouvé dans le dictionnaire.
- Si aucun objet similaire n'a été détecté, cela signifie qu'un nouvel objet a été détecté. Dans ce cas :
- Le nouvel ID est ajouté au dictionnaire "center_points" avec les coordonnées du centre calculées.
- La boîte englobante de l'objet ainsi que son ID sont ajoutés à la liste "objects_bbs_ids".
- La variable "id_count" est incrémentée de 1 pour le prochain ID.
- Après avoir traité tous les rectangles d'objets, un dictionnaire vide appelé "new_center_points" est créé.
- Une boucle est utilisée pour parcourir chaque objet dans la liste "objects_bbs_ids".
- L'ID de l'objet est extrait.
- Le centre correspondant à cet ID dans le dictionnaire "center_points" est récupéré.
- Le centre est ajouté au dictionnaire "new_center_points".
- Le dictionnaire "center_points" est mis à jour avec les nouveaux centres du dictionnaire "new_center_points".

- Finalement, la liste "objects_bbs_ids" est renvoyée.

En résumé, ce code permet de suivre des objets dans une scène en utilisant leurs boîtes englobantes et leurs positions centrales. Il maintient un dictionnaire des ID et des positions des objets déjà détectés, et associe les nouveaux objets détectés à des ID existants ou génère de nouveaux ID si nécessaire.

Notre code principal (main.py) utilise la bibliothèque OpenCV et PyTorch pour suivre les objets détectés dans une vidéo. Voici une interprétation détaillée de chaque bloc de code :

1. Import des bibliothèques :

- cv2 : Bibliothèque OpenCV pour la manipulation d'images et de vidéos.
- torch : Bibliothèque PyTorch pour le deep learning.
- numpy : Bibliothèque pour la manipulation de tableaux multidimensionnels.
- tracker : Un module personnalisé contenant la logique du suivi des objets.
- time : Module pour la mesure du temps.

2. Chargement du modèle YOLOv5 pré-entraîné :

- model = torch.hub.load('ultralytics/yolov5', 'yolov5s', pretrained=True) : Charge le modèle YOLOv5s pré-entraîné à l'aide de la fonction `load` de PyTorch Hub.

3. Ouverture de la vidéo en entrée :

- cap = cv2.VideoCapture('highway.mp4') : Ouvre la vidéo "highway.mp4" à l'aide de la fonction `VideoCapture` d'OpenCV.

4. Initialisation des variables et des structures de données :

- count = 0 : Variable pour suivre le nombre d'images traitées.
- tracker = Tracker() : Crée une instance du tracker d'objets.

- classNames = ["car", "motorcycle", "motorbike", "bus", "truck"] : Liste des noms de classes des objets d'intérêt.
- vehicles1 = {}`: Dictionnaire pour stocker les véhicules détectés dans la zone1.
- vehicles_t1 = {} : Dictionnaire pour stocker les temps de suivi des véhicules dans la zone1.
- vehicles2 = {} : Dictionnaire pour stocker les véhicules détectés dans la zone2.
- vehicles_t2 = {} : Dictionnaire pour stocker les temps de suivi des véhicules dans la zone2.
- temps_v1, speed_ms1, speed_kmh1 : Variables pour stocker le temps de suivi, la vitesse en m/s et la vitesse en km/h des véhicules dans la zone1.
- temps_v2, speed_ms2, speed_kmh2 : Variables pour stocker le temps de suivi, la vitesse en m/s et la vitesse en km/h des véhicules dans la zone2.
- b = "": Variable pour stocker le nom de la classe de l'objet en cours de suivi.

5. Définition de la fonction de rappel pour la souris :

- POINTS(event, x, y, flags, param) : Fonction de rappel appelée lorsqu'un événement souris se produit. Affiche les coordonnées BGR (Blue, Green, Red) du pixel survolé.

6. Configuration des zones de détection et de suivi :

- zone1 : Liste des coordonnées des points formant la zone1.
- zone_1 = set() : Ensemble vide pour stocker les identifiants des véhicules dans la zone1.
- var1, var2, var3, var4 : Variables pour extraire les coordonnées des points de la zone1.
- distance1 : Distance calculée à partir des coordonnées des points de la zone1.
- zone2 : Liste des coordonnées des points formant la zone2.

- var11, var22, var33, var44: Variables pour extraire les coordonnées des points de la zone2.
- distance2 : Distance calculée à partir des coordonnées des points de la zone2.
- zone_2 = set() : Ensemble vide pour stocker les identifiants des véhicules dans la zone2.

7. Boucle principale pour traiter chaque image de la vidéo :

- ret, frame = cap.read() : Lit une image de la vidéo. La variable `ret` indique si la lecture est réussie et `frame` contient l'image.
- if not ret: break : Si la lecture de l'image a échoué, la boucle est interrompue.
- count += 1 : Incrémente le compteur d'images.
- if count % 3 != 0: continue : Passe à l'itération suivante de la boucle si l'image actuelle n'est pas la troisième image (permet de réduire le nombre d'images traitées pour des raisons de performance).
- frame = cv2.resize(frame, (1020, 600)) : Redimensionne l'image à une taille de 1020x600 pixels.
- results = model(frame) : Effectue la détection d'objets sur l'image à l'aide du modèle YOLOv5.
- liste = [] : Liste vide pour stocker les coordonnées des boîtes englobantes des objets détectés.
- Boucle pour extraire les coordonnées des boîtes englobantes détectées :
- x, y, x1, y1: Coordonnées de la boîte englobante.
- b = str(rows['name']) : Extrait le nom de la classe de l'objet.\
- `liste.append([x, y, x1, y1])`: Ajoute les coordonnées à la liste `liste`.

- idx_box = tracker.update(liste) : Met à jour le suivi des objets à l'aide du tracker avec les nouvelles coordonnées des boîtes englobantes.
- Boucle pour dessiner les boîtes englobantes des objets suivis et calculer leur vitesse :
- x2, y2, x3 , y3 , id : Coordonnées et identifiant de la boîte englobante suivie.
- cv2.rectangle(frame, (x2, y2), (x3, y3), (0, 0, 255), 1) : Dessine un rectangle autour de la boîte englobante.
- Condition pour afficher le nom de la classe de l'objet ou uniquement l'identifiant.
- cv2.circle(frame, (x3, y3), 4, (0, 255, 0), -1) : Dessine un cercle à la position du coin inférieur droit de la boîte englobante.
- Calcule la distance entre le coin supérieur gauche et le coin inférieur gauche de la boîte englobante.
- Vérifie si la boîte englobante est dans la zone1 et met à jour les informations de suivi et de vitesse.
- Vérifie si la boîte englobante est dans la zone2 et met à jour les informations de suivi et de vitesse.
- cv2.polylines(frame, [np.array(zone1, np.int32)], True, (0, 255, 255), 3) : Dessine le contour de la zone1.
- cv2.polylines(frame, [np.array(zone2, np.int32)], True, (0, 255, 255), 3) : Dessine le contour de la zone2.
- Affiche le nombre de véhicules détectés dans la zone1 et la zone2.
- Affiche le nombre total de véhicules détectés.
- cv2.imshow("FRAME", frame) : Affiche l'image avec les annotations.
- if cv2.waitKey(0) & 0xFF == 27: break : Attends une pression de touche et vérifie si la touche échappement (ESC) est enfoncée pour sortir de la boucle.

8. Libération des ressources et fermeture des fenêtres :

- cap.release() : Libère la vidéo capturée.
- cv2.destroyAllWindows() : Ferme toutes les fenêtres d'affichage.

9. cap.release() : Libère la ressource vidéo en fermant la capture vidéo.

10. cv2.destroyAllWindows() : Ferme toutes les fenêtres d'affichage.

3.4) Limitations de travail :

Lors de la réalisation de notre projet de fin d'année nous avons été confrontés à différentes limitations. Ces limitations comprennent des contraintes temporelles.
Il est essentiel de prendre en compte ces limitations dès le début du projet afin de les gérer de manière appropriée et de maximiser les résultats malgré ces obstacles.

1. Limitation de temps : La réalisation d'un projet sur une période de deux mois impose des contraintes temporelles significatives. Durant ce laps de temps relativement court, il est essentiel de gérer efficacement chaque étape du processus. La recherche, la collecte de données, l'analyse, et le développement de l'application nécessitent tous une planification minutieuse pour respecter les délais impartis. La gestion du temps doit être rigoureuse, en attribuant des échéances précises à chaque phase du projet. Enfin, la capacité à s'adapter rapidement aux imprévus et aux obstacles est cruciale pour garantir le respect des délais de rendu du rapport final et du développement de l'application.

2. Limitations techniques : Selon la nature de votre projet, vous pourriez rencontrer des limitations techniques liées à des logiciels, des équipements ou des outils spécifiques dont vous avez besoin pour mener à bien votre projet.

3.5) Conclusion :

Le choix des technologies que nous venons de citer nous a beaucoup aidé dans la réalisation de notre projet et la minimisation du temps de travail, Django nous a fourni des fonctionnalités prédéfinies comme Django admin qui propose différentes fonctionnalités pour le rôle d'admin et la facilité de l'interaction avec la base de données, Bootstrap et Leaflet.js concernant l'interface graphique et l'utilisation de la map intégrée dans notre application.

Conclusion

En conclusion de notre projet de fin d'année, nous avons réussi à atteindre les buts et objectifs que nous avions fixés au début de notre étude pour résoudre notre problématique qui consiste à identifier, suivre et compter les véhicules dans leurs directions de déplacement respectives en utilisant une vidéo ou un flux de caméra. Nous avons mené une analyse approfondie de la situation étudiée, en nous basant sur des recherches et des méthodologies rigoureuses. Malgré les limitations et implications de notre travail, nos résultats fournissent une base solide pour des actions et des recommandations. Nous sommes fiers des résultats que nous avons obtenus et des efforts que nous avons déployés. Ce projet a été une occasion précieuse de mettre en pratique les connaissances et les compétences acquises tout au long de notre parcours scolaire. Nous avons fait face à des défis, des contraintes et des limites, mais nous les avons surmontés avec détermination et créativité. Nous avons travaillé en équipe, en tirant parti de nos forces individuelles pour accomplir des tâches complexes et atteindre nos objectifs communs. Ce projet nous a également permis de développer notre capacité à résoudre des problèmes, à prendre des décisions éclairées et à nous adapter aux imprévus.

Bibliographie

[1] :https://gurtam.com/fr/wialon?utm_campaign=WIALON_FR_SEARCH&utm_source=google&utm_medium=cpc&utm_content=591549999256&utm_term=syst%C3%A8me%20de%20g%C3%A9olocalisation%20de%20v%C3%A9hicule&gclid=CjwKCAjwjYKjBhB5EiwAiFdSfq6Zeg7aLVR96KlGH22J_H-QsuCREN0sb66ee7h77kAd5MqTnn0XXRoCwisQAvD_BwE
[2] : https://gurtam.com/fr/blog/wialon-data-processing
[3] : https://www.hikvision.com/fr/core-technologies/deep-learning/vehicle-identification/
[4] : https://www.hikvision.com/fr/core-technologies/
[5] : https://www.hikvision.com/fr/about-us/
[6] : https://cctvdirect.co.uk/blogs/industry-news/hikvision-review#:~:text=One%20reviewer%20wrote%2C%20%E2%80%9Cevery%20update,2021%20update%20to%20version%2014.15.
[7] : https://www.axopen.com/blog/2019/09/open-cv-cest-quoi/
[8] : https://datascientest.com/numpy
[9] : https://ledatascientist.com/debuter-avec-pytorch/#:~:text=PyTorch%20est%20un%20Framework%20Python,entra%C3%AEner%20des%20r%C3%A9seaux%20de%20neurones
[10] : https://medium.com/scalian/deep-sort-ou-comment-suivre-des-objets-dans-une-vid%C3%A9o-54ce98b59206
[11] : https://hashdork.com/fr/yolov5/
[12] : https://docs.python.org/fr/3.5/library/math.html
[13] : https://docs.python.org/fr/3/library/time.html

Annexe

- **Code de la classe tracker.py :**

```
import math

class Tracker:
    def __init__(self):
        self.center_points = {}
        self.id_count = 0
    def update(self, objects_rect):
        objects_bbs_ids = []
        for rect in objects_rect:
            x, y, w, h = rect
            cx = (x + x + w) // 2
            cy = (y + y + h) // 2
            same_object_detected = False
            for id, pt in self.center_points.items():
```

```
            dist = math.hypot(cx - pt[0], cy - pt[1])
            if dist < 35:
                self.center_points[id] = (cx, cy)
                objects_bbs_ids.append([x, y, w, h, id])
                same_object_detected = True
                break
        if same_object_detected is False:
            self.center_points[self.id_count] = (cx, cy)
            objects_bbs_ids.append([x, y, w, h, self.id_count])
            self.id_count += 1
    new_center_points = {}
    for obj_bb_id in objects_bbs_ids:
        _, _, _, _, object_id = obj_bb_id
        center = self.center_points[object_id]
        new_center_points[object_id] = center
    self.center_points = new_center_points.copy()
    return objects_bbs_ids
```

- **Code principal main.py :**

```
import cv2
```

```
import torch

import numpy as np

from tracker import *

import time

model = torch.hub.load('ultralytics/yolov5', 'yolov5s', pretrained=True)

cap = cv2.VideoCapture('highway.mp4')

count = 0

tracker = Tracker()

classNames = ["car", "motorcycle", "motorbike", "bus", "truck"]

vehicles1 = {}

vehicles_t1 = {}

vehicles2 = {}

vehicles_t2 = {}

temps_v1 = 0

speed_ms1 = 0

speed_kmh1 = 0

temps_v2 = 0

speed_ms2 = 0

speed_kmh2 = 0

b = ""
```

```
def POINTS(event, x, y, flags, param):
    if event == cv2.EVENT_MOUSEMOVE:
        colorsBGR = [x, y]
        print(colorsBGR)
cv2.namedWindow('FRAME')
cv2.setMouseCallback('FRAME', POINTS)
zone1 = [(365, 440), (329, 466), (526, 482), (533, 450)]
zone_1 = set()
var1 = zone1[0][0]
var2 = zone1[0][1]
var3 = zone1[1][0]
var4 = zone1[1][1]
distance1 = math.sqrt(pow((var3 - var1), 2) + pow((var4 - var2), 2))
zone2 = [(580, 427), (809, 408), (882, 455), (591, 474)]
var11 = zone2[0][0]
var22 = zone2[0][1]
var33 = zone2[3][0]
var44 = zone2[3][1]
distance2 = math.sqrt(pow((var33 - var11), 2) + pow((var44 - var22), 2))
zone_2 = set()
```

```
while True:
    ret, frame = cap.read()
    if not ret:
        break
    count += 1
    if count % 3 != 0:
        continue
    frame = cv2.resize(frame, (1020, 600))
    results = model(frame)
    liste = []
    for index, rows in results.pandas().xyxy[0].iterrows():
        print(rows)
        x = int(rows[0])
        y = int(rows[1])
        x1 = int(rows[2])
        y1 = int(rows[3])
        b = str(rows['name'])
        liste.append([x, y, x1, y1])
    idx_box = tracker.update(liste)
    for bbox in idx_box:
```

```
        x2, y2, x3, y3, id = bbox
        cv2.rectangle(frame, (x2, y2), (x3, y3), (0, 0, 255), 1)
        if b in classNames:
            cv2.putText(frame, b + str(id), (x2, y2), cv2.FONT_HERSHEY_PLAIN, 1.3, (255, 0,
0), 2)
        else:
            cv2.putText(frame, str(id), (x2, y2), cv2.FONT_HERSHEY_PLAIN, 1.3, (255, 0, 0),
2)
        cv2.circle(frame, (x3, y3), 4, (0, 255, 0), -1)
        res = cv2.pointPolygonTest(np.array(zone1, np.int32), (x3, y3), False)
        res1 = cv2.pointPolygonTest(np.array(zone2, np.int32), (x3, y3), False)
        if res > 0:
            if id not in zone_1:
                zone_1.add(id)
                vehicles1[id] = time.time()
        else:
            if id in zone_1:
                if id in vehicles1:
                    temps_v1 = time.time() - vehicles1[id]
                    if id in vehicles_t1:
                        temps_v1 = vehicles_t1[id]
```

```
            if id not in vehicles_t1:
                vehicles_t1[id] = temps_v1
            speed_ms1 = distance1 / temps_v1
            speed_kmh1 = speed_ms1 * 3.6
            if b in classNames:
                cv2.putText(frame, b + str(id) + " " + str(int(speed_kmh1)) + "km/h", (x2, y2),
                            cv2.FONT_HERSHEY_PLAIN, 1.3, (255, 0, 0), 2)
            else:
                cv2.putText(frame, str(id) + " " + str(int(speed_kmh1)) + "km/h", (x2, y2),
                            cv2.FONT_HERSHEY_PLAIN,1.3, (255, 0, 0), 2)
    if res1 > 0:
        if id not in zone_2:
            zone_2.add(id)
            vehicles2[id] = time.time()
    else:
        if id in zone_2:
            if id in vehicles2:
                temps_v2 = time.time() - vehicles2[id]
                if id in vehicles_t2:
                    temps_v2 = vehicles_t2[id]
```

```
            if id not in vehicles_t2:
                vehicles_t2[id] = temps_v2
        speed_ms2 = distance2 / temps_v2
        speed_kmh2 = speed_ms2 * 3.6
        if b in classNames:
            cv2.putText(frame, b + str(id) + " " + str(int(speed_kmh2)) + "km/h", (x2, y2),
                        cv2.FONT_HERSHEY_PLAIN, 1.3, (255, 0, 0), 2)
        else:
            cv2.putText(frame, str(id) + " " + str(int(speed_kmh2)) + "km/h", (x2, y2),
                        cv2.FONT_HERSHEY_PLAIN, 1.3, (255, 0, 0), 2)
cv2.polylines(frame, [np.array(zone1, np.int32)], True, (0, 255, 255), 3)
cv2.polylines(frame, [np.array(zone2, np.int32)], True, (0, 255, 255), 3)
print(zone_1)
z1 = len(zone_1)
txt = "Nb de vehicules : " + str(z1)
cv2.putText(frame, txt, (15, 192), cv2.FONT_HERSHEY_PLAIN, 1, (25, 0, 0), 2)
print(zone_2)
z2 = len(zone_2)
txt1 = "Nb de vehicules : " + str(z2)
cv2.putText(frame, txt1, (781, 188), cv2.FONT_HERSHEY_PLAIN, 1, (25, 0, 0), 2)
```

```
    txt2 = "Total : "+str(z1+z2)
    cv2.putText(frame, txt2, (457, 23), cv2.FONT_HERSHEY_COMPLEX, 0.7, (50, 136, 66),
2)
    cv2.imshow("FRAME", frame)
    if cv2.waitKey(0) & 0xFF == 27:
        break
cap.release()
cv2.destroyAllWindows()
```