REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEURE ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE ABOU -BEKR BELKAID - TLEMCEN

MEMOIRE

Présenté à :

FACULTE DES SCIENCES – DEPARTEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de :

MASTER EN INFORMATIQUE

Spécialité : Modèle intelligent et décision

Par :

Keche Walid

Ghezouti Meriem Chahrazed

Sur le thème

Feux Tricolores Intelligents :

Améliorer la Gestion du Trafic Urbain

Soutenu publiquement le 13 juin 2024 à Tlemcen devant le jury composé de :

Mme Labraoui Nabila	Université de Tlemcen	Présidente
Mme Seladji Yassamine	Université de Tlemcen	Encadrante
Mr Bouziane Youcef	Université de Tlemcen	Co-Encadrant
Mr Meziane Abdelfettah	Université de Tlemcen	Examinateur
Mr Etchiali Abdelhak	Université de Tlemcen	Expert i2e

Année Universitaire : 2023 ~ 2024

Dédicace

Nous dédions ce travail modeste a :

A nos chers parents, pour tous leurs sacrifices, leur amour,

Leurs soutiens et leurs prières tout au long de nos études,

A nos frères et sœurs, A nos amis.

Merci d'être toujours là pour nous.

Remerciements

Nous tenons tout d'abord à remercier notre Dieu qui nous a aidés, nous a donné la patience et le courage, et nous a accordé la santé et la volonté nécessaires pour compléter ce modeste travail.

Nous exprimons notre profonde gratitude envers Madame Seladji Yassamine pour sa guidance, son soutien constant et ses conseils avisés tout au long de la réalisation de ce projet. Son expertise et sa disponibilité ont été des atouts précieux qui ont grandement contribué à la réussite de ce mémoire.

Nous tenons également à exprimer notre profonde gratitude envers toutes les personnes qui nous ont soutenus et qui ont participé à la réalisation de ce mémoire, ainsi qu'à la réussite de cette année universitaire extraordinaire.

Nos remerciements s'adressent également aux membres du jury d'avoir accepté de lire et d'évaluer notre mémoire.

Merci à tous.

Table de matière

Liste des figures

Liste des tableaux

Introduction Générale

Introduction générale

Parmi les moyens de locomotion les plus utilisés pour assurer le déplacement d'un point à un autre, on compte les véhicules roulants sur des voies de communication conçues et réalisées pour remplir la fonction de chemins entre une provenance et une destination.

Les voies de communication, appelées plus communément routes, étant exploitées par plusieurs utilisateurs à la fois, nécessitent une règlementation pour assurer la sécurité de ses usagers en accord avec une utilisation universelle. Cette réglementation s'appelle « code de la route » [1].

Le code de la route regroupe des règles et des réglementations mises en place par les autorités gouvernementales afin de réguler la circulation des véhicules sur les routes publiques. Son objectif est de garantir la sécurité des utilisateurs de la route, de contrôler le trafic routier et de prévenir les accidents. Ces règles concernent différents aspects de la conduite, comme les limites de vitesse, les règles de priorité, les signaux de circulation, les règles de stationnement et bien d'autres encore.

Contexte :

Notre étude se concentre spécifiquement sur la régulation du trafic aux intersections, des points cruciaux où la coordination des mouvements des véhicules est vitale. Pour ce faire, les autorités compétentes ont mis en place diverses mesures régulatrices, telles que les panneaux de stop, de cédez-le-passage, les règles de priorité et les feux tricolores. Cependant, ces règlements présentent une limitation majeure : leur caractère statique. En effet, ils ne s'adaptent pas à la fluctuation de la densité du trafic, sauf dans le cas des feux tricolores dotés d'un contrôle automatique.

Les feux tricolores préprogrammés peuvent entraîner des difficultés dans la gestion du trafic, notamment lorsque des zones dégagées affichent le feu vert tandis que des zones congestionnées sont contraintes d'attendre au feu rouge. Bien que l'intervention humaine puisse temporairement résoudre ce problème, elle s'avère souvent insuffisante dans des conditions extrêmes telles que les fortes chaleurs estivales, les heures tardives de la nuit ou les intempéries

hivernales. C'est dans ce contexte que s'impose l'idée d'intégrer un système de contrôle adaptatif des feux tricolores, connu sous le nom de feux tricolores intelligents.

Objectifs :

L'objectif principal de ce projet de fin d'études de master est la conception et la réalisation d'un système de contrôle de feux de circulation intelligent, visant à fluidifier le trafic et à atténuer les problèmes de congestion aux intersections. Nous cherchons à explorer les avantages potentiels des feux tricolores intelligents dans ce contexte urbain en constante évolution.

+ Le présent mémoire est organisé en quatre chapitres comme suit :

1. Le premier chapitre explore les défis et les solutions liés à la gestion du trafic urbain aux intersections.
2. Le chapitre suivant aborde le problème de la détection des objets en temps réel. Nous y décrirons en particulier l'algorithme Yolo, basé sur le deep learning, qui est une référence dans le domaine.
3. La conception et la mise en œuvre du projet sont présentées dans le troisième chapitre, qui décrit les outils utilisés pour cette mise en œuvre.
4. Dans le quatrième chapitre, l'implémentation et les tests de validation du fonctionnement du système réalisé sont exposés.
5. A la fin du mémoire, nous tirerons les principales conclusions concernant le travail accompli dans le cadre de ce projet et nous formulerons des recommandations pour d'éventuelles études futures qui viendront compléter et améliorer ce modeste travail.

Chapitre I Étude des défis et solutions de la gestion du trafic urbain aux intersections

1. Introduction

L'espace urbain est un terrain partagé entre divers usagers incluant les conducteurs de voitures, les usages des transports en commun, les piétons, et autres. Avec la croissance rapide des espaces urbains le nombre de véhicules a considérablement augmenté, ce qui a causé des problèmes de congestion dans les villes et les agglomérations [3].

Les problèmes majeurs causés par la congestion, en particulier dans les intersections urbaines, sont des retards fréquents dans les transports en commun, une détérioration de la qualité de l'air due à la pollution, le stress, et autres [4].

Pour faire face à ces problèmes, de nouvelles méthodes de gestion du trafic ont vu le jour. Ces méthodes s'inscrivent dans le cadre des "Systèmes du Trafic Intelligent" (STI). Ces systèmes utilisent des technologies avancées pour optimiser la circulation routière, réduire le temps d'attente, et améliorer la sécurité des usagers [5].

Dans ce chapitre, nous nous pencherons sur les fondements des intersections urbaines et les défis associés à leur gestion. Nous examinerons plus en détail les règles qui régissent ces intersections, en mettant l'accent sur les principes de priorité et de sécurité qui orientent le trafic. Le but est de donner une vision approfondie des systèmes de gestion du trafic dans un contexte urbain en constante évolution [6].

2. Gestion du trafic urbain

L’infrastructure urbaine est partagée entre des différents usagers. Il est donc indispensable de procéder à la gestion du trafic urbain afin d’éviter des situations de blocage, qui peuvent avoir des conséquences négatives sur les usagers de la route et sur l’environnement.

2.1.Facteurs contribuant à la congestion routière

La congestion urbaine survient lorsque la capacité de l'infrastructure est dépassée par le nombre de véhicules utilisés, comme le montre la figure 1. Deux catégories d'encombrement sont identifiées en centre-ville : la congestion récurrente, qui se produit pendant les heures de pointe ou lors d'événements spéciaux, et la congestion non récurrente, qui résulte de travaux ou d'accidents routiers.

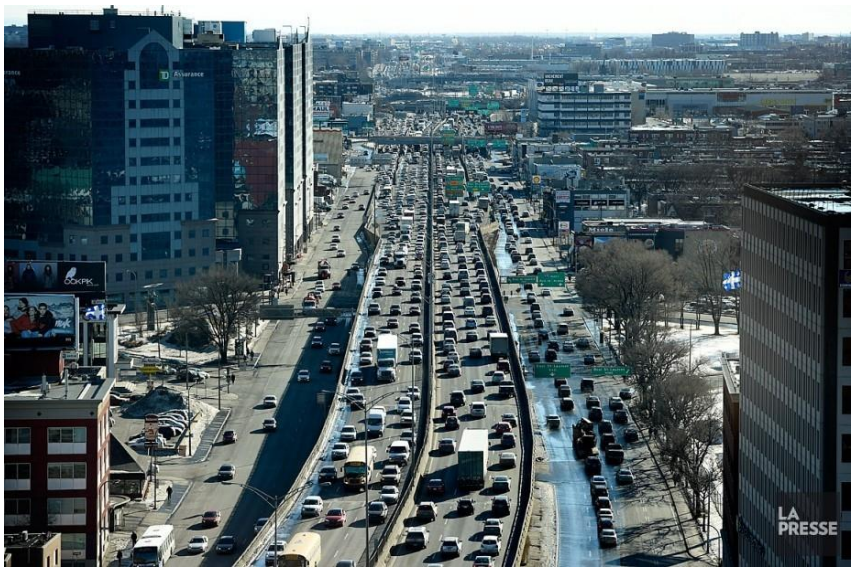


Figure 1. Congestion routière

2.2.Conséquences de la congestion sur l'environnement et la mobilité urbaine

La congestion des zones urbaines a des répercussions importantes sur les villes, impactant à la fois l'environnement et la mobilité. Voici quelques exemples de ces répercussions :

1. L'Émissions de CO_2 et la pollution d'air.
2. La congestion accélère l'usure des routes et nécessite des réparations fréquentes.
3. Les embouteillages entraînent une perte de temps pour les conducteurs et les passagers, affectant la productivité.
4. Être coincé dans les embouteillages peut entraîner du stress et de la frustration chez les conducteurs.

2.3.Méthodes de réduction de la congestion routière

Parmi les méthodes de réduction de la congestion routière, on peut citer l'utilisation de signalisation routière, la régulation de la circulation par des agents de police, l'aiguillage du trafic, l'affectation dynamique des voies et l'utilisation des feux tricolores. Les trois dernières méthodes peuvent être automatisées. Comme déjà stipulé, dans ce travail nous nous intéressons à l'automatisation des feux tricolores. Plus exactement, nous essayons de concevoir et réaliser des feux dits intelligents, c'est-à-dire des feux dont le réglage des cycles s'adapte à la densité du trafic sur les différentes voies.

2.4.Gestion du trafic au niveau des intersections urbaines

2.4.1. Définition des intersections urbaines

Les intersections urbaines sont des endroits où se croisent plusieurs voies, souvent formant un carrefour. Elles jouent un rôle central dans le trafic urbain, où les voitures, les piétons et parfois les cyclistes doivent s'organiser pour passer en toute sécurité.

Les intersections peuvent présenter différentes caractéristiques, allant des croisements simples où deux routes se croisent aux échangeurs complexes. L'endroit où se croisent les voitures s'appelle "La zone de conflit", comme le montre la figure 2.

Elles sont souvent munies de dispositifs de contrôle comme les feux de circulation afin de réguler le trafic et garantir une circulation fluide.

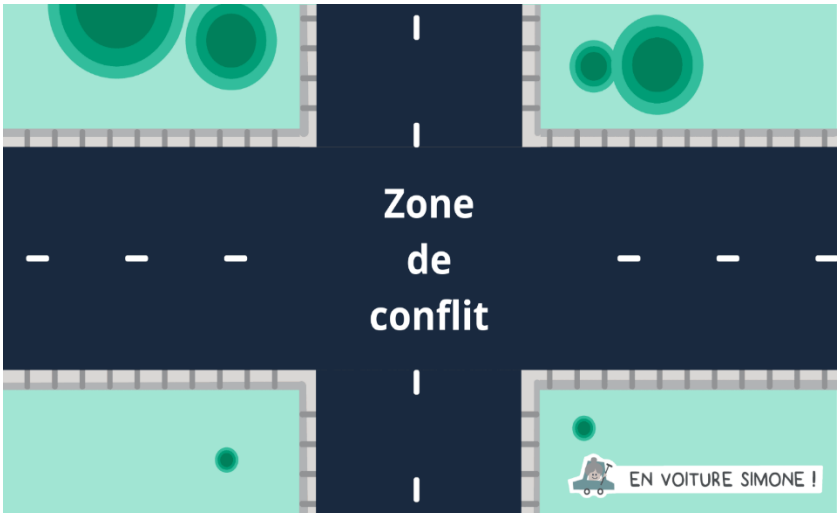


Figure 2. Zone d'une intersection

2.4.2. Règles générales de gestion des intersections urbaines

Les priorités de passage sont définies dans les règles de gestion des intersections urbaines, qui déterminent le véhicule ou le piéton qui doit passer en premier. Les règles de signalisation, comme les feux de circulation et les panneaux de signalisation, sont également incluses dans celles-ci afin de guider et de sécuriser la circulation. Finalement, ces règles incluent les consignes de conduite propres à chaque type d'intersection, comme les consignes de virage et de changement de voie.

3. La gestion par les feux de signalisation tricolores :

Une intersection éclairée est une intersection où le trafic est régulé par des feux de signalisation lumineux pilotés par un compteur. Le réglage des cycles de feux doit garantir la sécurité des véhicules et des piétons tout en favorisant une circulation fluide.

3.1. Historique

Les feux tricolores ont vu le jour en 1914. Bien que le premier feu rouge ait été érigé à Londres en 1868, consistant en une lanterne à gaz arborant le rouge pour "arrêt" et le vert pour "attention", le premier feu rouge électrique a été implanté à Cleveland, aux États-Unis, en 1914, comme le dépeint la figure 3. Ce dispositif était facilement repérable de loin et fonctionnait de manière assez rudimentaire : le rouge indiquait l'arrêt, tandis qu'une sonnerie marquait le départ. L'inventeur de ce feu électrique était Garrett Augustus Morgan, également connu pour avoir créé le masque à gaz utilisé pendant la Grande Guerre.

En 1923, un autre feu rouge électrique fut mis en place à Paris, au croisement des boulevards Saint-Denis et Sébastopol. Depuis lors, les feux tricolores que nous connaissons aujourd'hui, munis de leurs couleurs verte, orange et rouge, se sont répandus à travers le monde pour réguler la circulation routière, ferroviaire et fluviale en raison de leur importance.

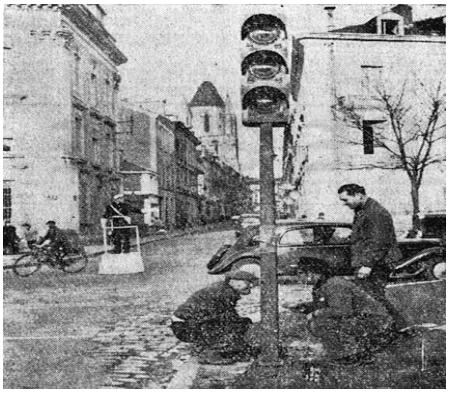


Figure 3. Ancienne photo d’un feu de signalisation tricolore.

3.2. Feux tricolores traditionnels

Les traditionnels feux tricolores reposent sur un système de temporisation fixe, avec parfois quelques options telles que les feux de dégagement et quelques options pour les piétons. Ils accordent un ordre de priorité en fonction de la teinte affichée : Orange, Vert et Rouge.

+ **Feu Rouge** : Impose au conducteur de marquer l'arrêt du véhicule, comme illustré dans la figure 4.

Figure 4. Feu Rouge

+ **Feu Orange** : Signale le passage du rouge au vert, incitant les conducteurs à ralentir et à s'arrêter avant que le feu ne devienne rouge, tel qu'illustré dans la figure 5..

Figure 5. Feu Orange

+ **Feu vert** : Alerte les conducteurs qu'ils peuvent progresser en toute sécurité à travers l'intersection, comme illustré dans la figure 6.

Figure 6. Feu vert

3.3. Système tricolore traditionnel avec une variation de temps par période

En fonction du trafic sur les différentes voies, les durées d'allumage des feux tricolores sont adaptées en se basant sur des mesures statistiques du débit de véhicules pour chaque voie. Le trafic est dense le matin et en fin d'après-midi pendant les heures de pointe, en fonction des activités diverses dans la ville. Le trafic fluctue généralement en fonction des directions tout au long de la journée.

4. Feux tricolores intelligents

Un système de contrôle de la circulation routière appelé feu tricolore intelligent utilise des technologies de pointe pour réguler le trafic de manière adaptative et performante. Sa structure globale est composée de plusieurs éléments essentiels :

+ **Capteurs de trafic : L**es appareils tels que les Boucles électromagnétiques, Capteurs vidéo (caméra intelligente), Réseaux de capteurs sans fil, Capteurs à hyperfréquence (radar) et Fibres optiques servent à recueillir en temps réel des informations sur le trafic, comme le nombre de véhicules, la vitesse et la densité de la circulation [9].
+ **Système de traitement des données :** Un système informatique analyse les données recueillies par les capteurs en utilisant des algorithmes avancés pour interpréter les conditions de circulation et prendre des décisions concernant la synchronisation des feux de signalisation [10].
+ **Contrôleur de feux de signalisation :** Ce composant a pour mission de contrôler les feux de signalisation en se basant sur les données fournies par le système de traitement des données. Il calcule la durée des périodes de feu en fonction du trafic afin d'améliorer la fluidité du trafic.
+ **Communication en réseau :** Le réseau de communication entre les feux tricolores intelligents permet souvent aux différents éléments du système de partager des informations en temps réel et de coordonner leurs actions afin de gérer efficacement le trafic.
+ **Interface utilisateur :** Certains systèmes peuvent également inclure une interface utilisateur, telle qu'une application mobile ou un panneau d'affichage, permettant aux

autorités de contrôle de surveiller et de gérer le système, ainsi que de communiquer des informations importantes aux usagers de la route.

En collaboration, ces éléments permettent aux feux tricolores intelligents de s'ajuster de manière dynamique aux variations des conditions de circulation, ce qui améliore la fluidité du trafic, réduit les temps d'attente et contribue à la sécurité routière.

4.1. Etat de l'art

Dans l'état de l'art des feux tricolores intelligents, plusieurs systèmes innovants sont déployés à travers le monde pour améliorer la gestion du trafic. Voici trois exemples :

+Feux de circulation adaptatifs (FCA) : Ces systèmes utilisent des capteurs et des algorithmes avancés pour ajuster dynamiquement la durée des feux de signalisation en fonction du flux de trafic en temps réel. Ils peuvent détecter les niveaux de congestion et optimiser les temps de signalisation pour minimiser les retards et les temps d'attente.

+Feux de circulation connectés : Ces feux sont intégrés à un réseau de communication intelligent, leur permettant de communiquer avec les véhicules équipés de technologies de communication de véhicule à infrastructure (V2I). Les feux peuvent ainsi transmettre des informations sur leur état actuel, permettant aux véhicules de s'adapter en conséquence, comme ajuster leur vitesse pour éviter de passer au rouge.

+Feux de circulation basés sur l'apprentissage automatique : Ces systèmes utilisent des techniques d'apprentissage automatique pour analyser les modèles de trafic et prédire les flux de véhicules. En fonction de ces prédictions, les feux peuvent ajuster leurs cycles de signalisation pour optimiser le flux de trafic et réduire les temps d'attente.

+Feux de circulation à priorité pour les transports en commun : Dans certaines villes, des systèmes de feux de circulation sont mis en place pour donner la priorité aux transports en commun, tels que les bus et les tramways. Ces feux peuvent détecter l'approche des véhicules de transport en commun et ajuster leur cycle de signalisation pour minimiser les retards et améliorer l'efficacité des transports publics.

4.2. Exemples de Systèmes de Feux Tricolores Intelligents dans le Monde

1. Système de Feux Tricolores Intelligents à Singapour

- Description : À Singapour, la Land Transport Authority (LTA) a mis en place des feux tricolores intelligents capables de s'adapter en temps réel aux conditions de circulation. Ces systèmes utilisent des caméras et des capteurs pour surveiller le flux de trafic et ajuster les temps de feu en conséquence.
- Fonctionnalités :
    - Surveillance en temps réel du trafic.
    - Ajustement dynamique des cycles de feux pour minimiser les temps d'attente.
    - Utilisation de l'intelligence artificielle pour prédire les pics de trafic.

2. Système SCATS en Australie

- Description : Le Sydney Coordinated Adaptive Traffic System (SCATS) est un système de gestion du trafic en temps réel développé en Australie. SCATS ajuste les feux tricolores en fonction des données de trafic collectées à partir de capteurs et de caméras.
- Fonctionnalités :
    - Ajustement automatique des cycles de feux pour optimiser le flux de trafic.
    - Collecte de données à partir de capteurs installés dans la chaussée.
    - Utilisation d'algorithmes adaptatifs pour gérer le trafic de manière efficace.

3. Système SCOOT au Royaume-Uni

- Description : Le Split Cycle Offset Optimization Technique (SCOOT) est un système de gestion du trafic en temps réel utilisé dans plusieurs villes du Royaume-Uni. SCOOT utilise des capteurs pour surveiller le trafic et ajuste les feux tricolores en conséquence.
- Fonctionnalités :
    - Optimisation en temps réel des temps de feu pour réduire les embouteillages.
    - Collecte de données à partir de boucles de détection enfouies dans la chaussée.
    - Coordination des feux sur les routes principales pour assurer un flux de trafic continu.

4. Feux Tricolores Intelligents à Barcelone, Espagne

- Description : Barcelone a intégré des feux tricolores intelligents dans son système de gestion du trafic urbain. Ces feux utilisent des capteurs et des caméras pour ajuster les cycles de feux en fonction du volume de trafic.
- Fonctionnalités :
    - Utilisation de caméras et de capteurs pour détecter les véhicules.
    - Ajustement des temps de feu en temps réel pour fluidifier le trafic.
    - Intégration avec d'autres systèmes de gestion de la mobilité urbaine pour une coordination optimale.

Les exemples de systèmes précédemment fournis sont des systèmes développés à l'étranger. Chaque système est conçu pour s'adapter aux spécificités de la circulation et de la gestion du trafic de son pays d'origine. Si l'Algérie souhaite les acheter, ils seront très coûteux. C'est pourquoi nous avons proposé notre propre projet, afin de l'adapter au marché algérien.

5. Notre Feux Tricolores Intelligents Proposés

- Description : Le projet de feux tricolores intelligents intègre un microprocesseur et une caméra pour détecter les véhicules et ajuster les cycles de feux en fonction du volume de trafic dans chaque zone. Ce système vise à optimiser la gestion du trafic urbain en temps réel.
- Fonctionnalités :
    - Utilisation de caméras pour détecter les véhicules : La caméra capture des images en temps réel, permettant au système de compter les véhicules présents dans chaque zone de l'intersection.
    - Ajustement des temps de feu en temps réel : Le microprocesseur analyse les données collectées et prend des décisions basées sur le nombre de véhicules détectés. Cela permet d'ajuster les durées des feux tricolores pour minimiser les temps d'attente et fluidifier le trafic.
    - Contrôle des LEDs : Le microprocesseur contrôle les LEDs des feux tricolores en fonction des décisions prises, assurant ainsi une gestion dynamique et efficace du trafic.
    - Extension potentielle : Intégration avec d'autres systèmes de gestion de la mobilité urbaine pour une coordination optimale, incluant des systèmes de transport public et des applications de navigation en temps réel.

4.3. Limitation et défis de l'implémentation des feux tricolores intelligents

La mise en place des feux tricolores intelligents fait face à différents défis et contraintes, parmi lesquels la complexité de l'environnement urbain est une préoccupation majeure [10]. Les systèmes de contrôle de la circulation doivent être parfaitement adaptables à la dynamique des interactions entre les usagers de la route (véhicules, piétons, cyclistes) et aux structures diverses des intersections urbaines. La collecte de données précises et la prise de décisions en temps réel sont indispensables pour assurer l'efficacité des feux tricolores intelligents.

De plus, les aspects de sécurité et de fiabilité représentent un autre défi important lors de la mise en place de ces systèmes [9]. Il est primordial de garantir la sécurité des utilisateurs de la route dans un contexte urbain dynamique et souvent imprévisible. Il est essentiel de concevoir et de mettre en place des feux tricolores intelligents en respectant des normes de sécurité rigoureuses afin de prévenir les accidents et les incidents. En outre, il est essentiel de garantir la fiabilité des systèmes afin de garantir la fluidité de la circulation et d'éviter les pannes potentiellement nocives. Pour assurer le bon fonctionnement des feux tricolores intelligents dans toutes les conditions, il est essentiel d'avoir une conception solide, des tests approfondis et une maintenance régulière.

5. Conclusion

En résumé, ce chapitre a présenté les principes essentiels et les défis associés à la gestion du trafic urbain, en mettant l'accent sur l'importance des intersections urbaines et des feux tricolores pour cette gestion. Nous nous sommes penchés sur l'évolution historique des feux de signalisation, des modèles classiques aux solutions intelligentes, tout en signalant les contraintes inhérentes à leur mise en place, telles que la complexité de l'environnement urbain et les aspects de sécurité et de robustesse. Il est essentiel de prendre en considération ces contraintes afin de concevoir des systèmes de gestion du trafic efficaces et adaptatifs, qui favorisent l'amélioration de la mobilité urbaine et assurent la sécurité des usagers de la route.

Chapitre II : Détection des objets en temps réel.

1. Introduction à la Détection d'Objets en Temps Réel

1.1.Définition

La détection d'objets en temps réel est une composante essentielle de la recherche en vision par ordinateur, se situant à l'intersection de deux autres domaines : la classification d'images et la localisation d'objets. Elle consiste en l'identification et le suivi d'objets en mouvement dans une scène en temps réel.

La réussite de la détection d'objets repose sur un algorithme solide de classification d'images. Le deep learning, en particulier les réseaux convolutionnels, est devenu la méthode privilégiée pour résoudre ce type de problèmes depuis les résultats du challenge ImageNet en 2012 [11].

1.2.Domaines d'Application

La détection d'objets en temps réel peut être utilisée dans différentes applications, comme :

- **La sécurité :** Elle est utilisée dans la surveillance vidéo pour détecter les intrusions [12] et les activités suspectes comme illustré dans la figure 7.

Figure 7. Algorithme et IA pour sécuriser les commerces

- **L'automobile** : Elle est employée dans la sécurité routière où elle est essentielle pour la détection des piétons, des panneaux de signalisations, et des véhicules comme le montre la figure 8.

Figure 8. Segmentation et détection d'objets en temps réel

- **L'industrie :** Elle est utilisée pour optimiser les processus industriels : contrôle de qualité, suivi d'objets sur une chaine de production, etc.

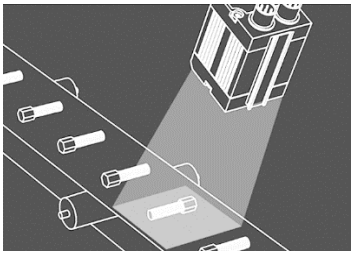


Figure 9. Capteur de détection pour la présence et le contrôle de forme

- **La médecine :** détection de tumeurs, analyse IRM, et d'autres applications médicales. [14].

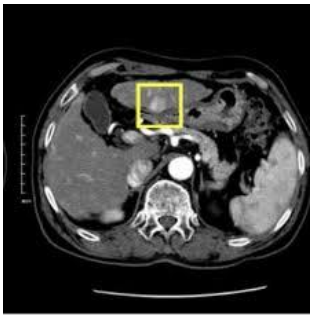


Figure 10. Détection et segmentation en imagerie médicale

En résumé, la détection d'objets en temps réel permet une réponse immédiate aux événements en cours, améliorant ainsi la performance des applications dans de nombreux domaines.

2. Utilisation du Deep Learning pour la détection d'objets en temps réel

La détection d'objets se faisait initialement avec des méthodes classiques. Depuis quelques temps, elle se fait de plus en plus avec des techniques nouvelles, en particulier le deep learning, qui permettent d'améliorer la qualité de cette détection.

2.1. Deep Learning

Le concept d'apprentissage constitue un pilier fondamental de l'intelligence artificielle (IA), conférant aux machines la capacité de détecter des objets en s'appuyant sur leurs expériences de détection passées. Son objectif est de saisir la structure des données et de les représenter de manière intelligible et fonctionnelle. Généralement, il existe deux types d'apprentissage : supervisé et non supervisé [26].

Le Deep Learning (DL), appelé apprentissage profond en Français, est l'une des techniques majeures du Machine Learning (ML). Comme le montre la Figure 11, Il fait appel à des réseaux de neurones afin d'obtenir des modèles spécifiques. Ces réseaux s'inspirent de la structure et de l'organisation du système cérébral humain, permettant ainsi aux systèmes d'intelligence artificielle d'accomplir des tâches humaines telles que la reconnaissance visuelle d'objets réels ou la compréhension de la parole. Le DL constitue une catégorie d'algorithmes de ML [26].

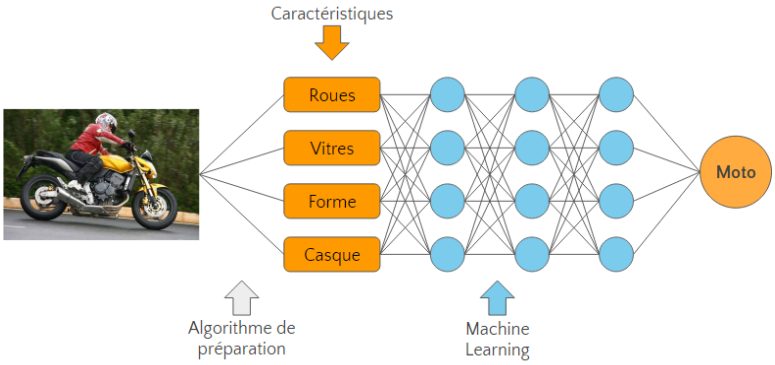


Figure 11. Concept du Deep Learning

Les réseaux de neurones utilisés dans le DL sont organisés en couches. Les informations sont transmises à travers chaque couche, la sortie de la couche précédente servant d'entrée pour la couche suivante. La première couche du réseau est appelée couche d'entrée, tandis que la dernière est la couche de sortie. Toutes les couches intermédiaires sont désignées comme des couches cachées. Avec l'augmentation du nombre de neurones, le réseau devient plus "profond". Chaque couche utilise généralement un algorithme simple et homogène comprenant une fonction d'activation d'un certain type [26].

L'apprentissage profond est employé dans divers secteurs, tels que :

+ La détections de fraudes.
+ Le service client.
+ Les véhicules autonomes.
+ Le traitement du langage naturel.

2.2. Réseaux Neuronaux Convolutifs (CNN)

2.2.1. Définition

Les réseaux neuronaux convolutifs (CNN), aussi connus sous le nom de ConvNets, sont un genre de réseaux neuronaux à anticipation qui sont particulièrement adaptés aux tâches liées à la vision par ordinateur, en particulier la reconnaissance d'objets. Les CNN sont à ce jour les modèles les plus performants pour classer des images.

Le fonctionnement des réseaux neuronaux convolutifs repose sur l'ingestion et le traitement de grandes quantités de données dans un format de grille, puis l'extraction de caractéristiques granulaires essentielles pour la classification et la détection. En général, les CNN sont constitués de trois couches distinctes : une couche convolutive, une couche de mise en commun et une couche entièrement connectée. Chaque couche vise un but précis., réalise une tâche sur les informations ingérées et acquiert des connaissances de plus en plus complexes [26].

2.2.2. Architecture des Réseaux Neuronaux Convolutifs

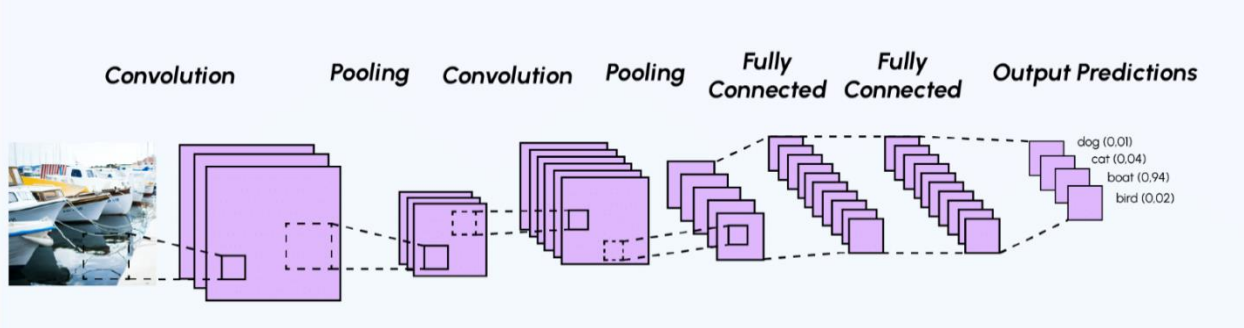


Figure 12. Réseau convolutif neuronal

Comme illustré à la figure ci-dessus, une architecture CNN est constituée d'un empilement de différentes couches de traitement [26] :

+ La couche de convolution (CONV) qui gère les données provenant d'un champ récepteur.
+ L'utilisation de la couche de pooling (POOL) permet de réduire la taille de l'image intermédiaire en utilisant souvent un sous-échantillonnage.
+ La couche de correction (ReLU) en référence à la fonction d'activation (Unité de rectification linéaire).
+ La couche de type perceptron est la couche "entièrement connectée" (FC).
+ La couche de perte (LOSS).

a. **La couche Convolution :**

On peut parfois désigner une couche convolutive comme une couche d'extraction de caractéristiques, car c'est à ce niveau que les caractéristiques de l'image sont extraites.

En premier lieu, la couche Convolution est reliée à une partie de l'image pour effectuer l'opération de convolution. Le résultat de cette opération est le produit scalaire du champ récepteur (qui est de la même taille que le filtre dans l'image d'entrée), et le filtre est l'unique entier de la sortie. Le filtre est ensuite déplacé vers le champ récepteur suivant de la même image d'entrée et nous répétons la même opération.On répète cette opération à plusieurs reprises selon le même processus jusqu'à ce que l'image entière soit générée en numérisation [27].

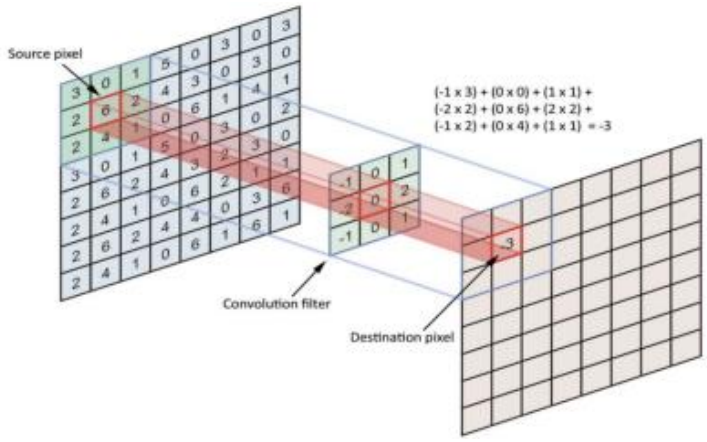


Figure 13. Couche de convolution

b. **Couche de mise en commun (Pooling) :**

Une opération de sous-échantillonnage appelée couche de regroupement (POOL) est souvent utilisée entre deux couches convolutives.

Sa fonction consiste à diminuer progressivement la taille de la carte des caractéristiques (matrice de convolution) afin de diminuer les paramètres et les calculs du réseau, tout en préservant les informations essentielles [27].

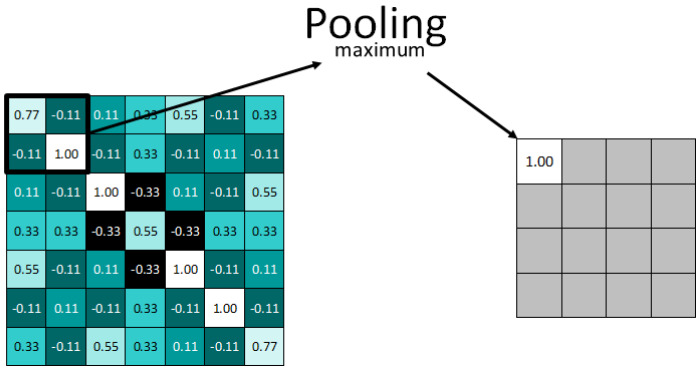


Figure 14. Couche Pooling

c. **La couche entièrement connectée (Fully-Connected)**

Toutes les entrées d'une couche dans un réseau neuronal sont connectées à chaque fonction d'activation de la couche suivante, ce qui en fait une couche entièrement connectée.Au sein d'un modèle d'apprentissage automatique, la couche finale est une couche entièrement

connectée qui exploite les informations extraites par la couche précédente afin de générer la sortie finale.[27]

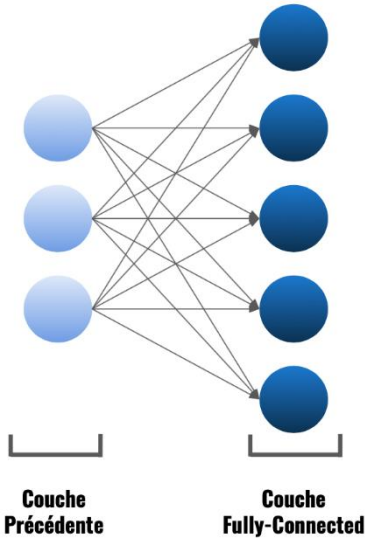


Figure 15. Couche entièrement connectée

2.3. Modèles de détection d'objets en temps réel

2.3.1. Modèles basés sur le deep learning

Ces dernières années, la détection d'objets basée sur le deep learning a fait des progrès significatifs par rapport aux méthodes traditionnelles. Cette section décrit les méthodes de détection les plus couramment utilisées : Fast R-CNN, RetinaNet, SSD (MultiBox Single Detector) et YOLO (You Look Only Once).

- ✓ **Faster R-CNNs [18] :**

Faster R-CNN est l'un des algorithmes populaires et performants dans la détection d'objets. Il fait partie de la série R-CNN, développée par Ross Girshick et son équipe en 2014. Améliorée avec Fast R-CNN, elle a ensuite abouti à Faster R-CNN [21]. Il est basé sur les réseaux de neurones convolutifs (CNN).

Le modèle Faster R-CNN commence par l'extraction des fonctionnalités à l'aide d'un réseau neuronal convolutif (CNN). Ensuite, un module de proposition de régions (RPN) est utilisé pour générer des propositions de régions d'intérêt (ROI). Enfin, ces propositions sont classées et affinées pour détecter et localiser des objets dans les images [18].

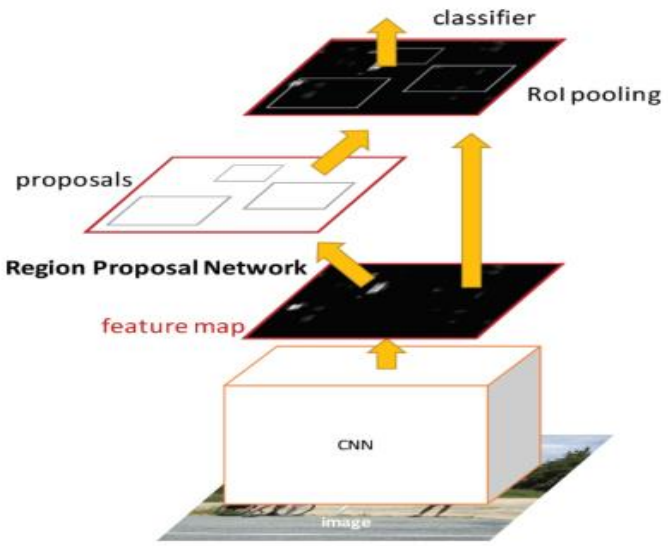


Figure 16. Architecture du modèle faster R-CNN [18].

- ✓ **SSD (Single Shot MultiBox Detector) [19] :**

SSD est un autre algorithme de détection d'objets basé sur des réseaux de neurones convolutifs. Contrairement à Faster R-CNN, SSD effectue la détection et la classification des objets en un seul passage, ce qui entraîne un traitement en temps réel plus rapide et plus efficace. Il détecte des objets de différentes tailles et formes à l'aide de plusieurs échelles de prédiction. Enfin, il applique un cadre de délimitation prédéfini pour rechercher et classer les objets dans l'image [19].

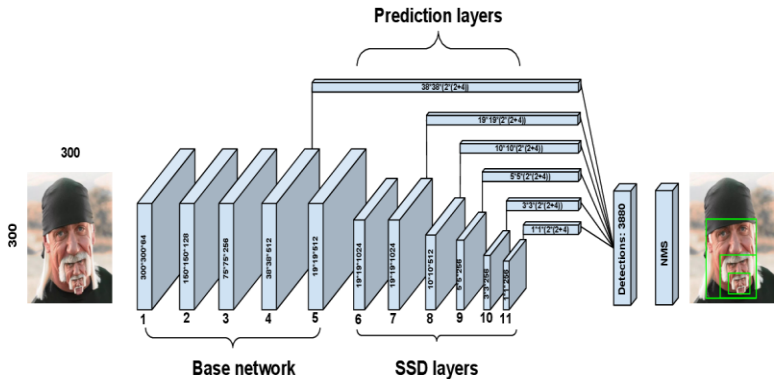


Figure 17. Schéma d'un SSD

✓ **RetinaNet [20] :**

RetinaNet est un algorithme de détection d'objets conçu pour résoudre le problème de la répartition déséquilibrée des classes dans les données d'entraînement. Il utilise une architecture de réseau pyramidal pour détecter des objets à **différentes** échelles et introduit une nouvelle perte appelée perte focale pour mieux gérer les exemples complexes.

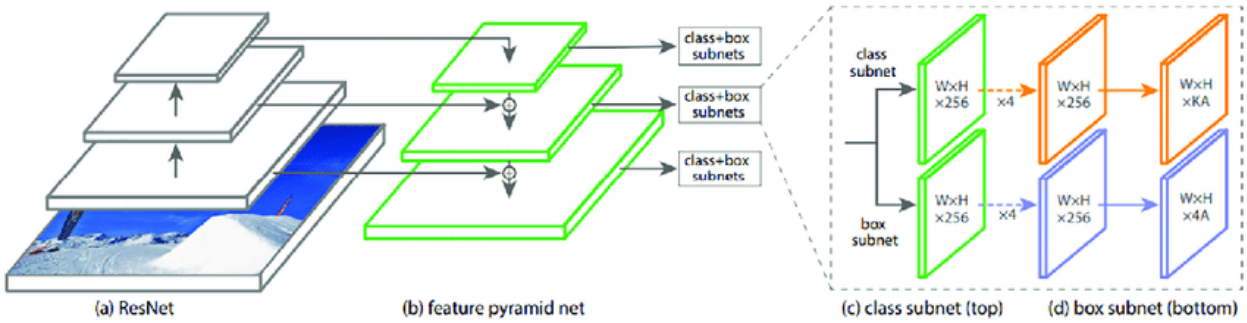


Figure 18. Schéma d'un RetinaNet

✓ **Modèle YOLO (You Only Look Once)**

Dans la vision par ordinateur, la détection d'objet a été réinventée par l'algorithme révolutionnaire YOLO (You Only Look Once), qui a ouvert une nouvelle perspective dans la détection d'objets. Son impact est visible dans de nombreux domaines, de la surveillance à la conduite autonome. C'est ce modèle que nous avons adopté dans le présent travail. Nous allons le détailler dans ce suit.

2.4. Description détaillée du modèle YOLO

2.4.1. Définition

You Only Look Once (YOLO) est un algorithme de détection d'objets en temps réel de pointe introduit en 2015 par Joseph Redmon , Santosh Divvala , Ross Girshick et Ali Farhadi dans leur célèbre article de recherche « You Only Look Once : Détection d'objets unifiée et en temps réel » [22].

YOLO repose sur des réseaux de neurones convolutifs (CNN) [13]. Contrairement aux méthodes traditionnelles, YOLO effectue la détection et la classification des objets en une seule passe dans l'image, ce qui garantit un traitement rapide et efficace en temps réel [13]. Il est

capable de détecter et de localiser simultanément plusieurs objets dans une image grâce à l'utilisation de grilles prédictives et de champs d'ancrage [13].

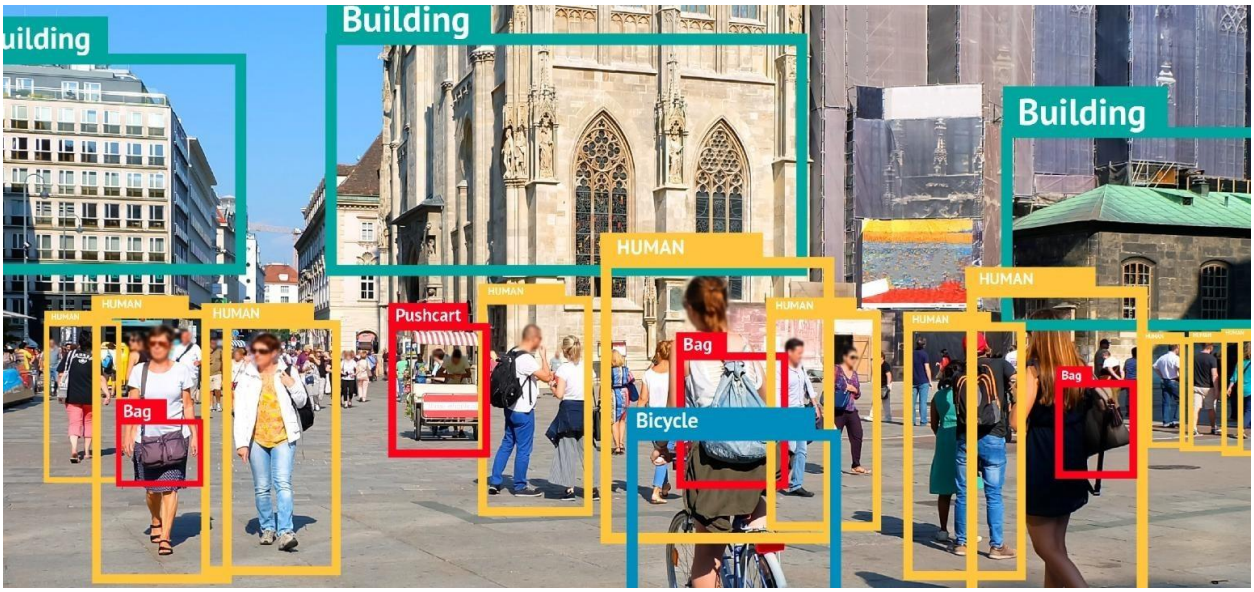


Figure 19. Exemple de résultat de détection avec YOLO

2.4.2. Caractéristiques du Modèle YOLO

Les caractéristiques suivantes démontrent les atouts significatifs du modèle YOLO dans le domaine de la détection d'objets en temps réel :

+ **Vitesse** : Yolo se distingue par sa vitesse de détection en temps réel. Grâce à son architecture efficiente, il est capable de traiter les images rapidement [13].
+ **Précision de détection** : Malgré la vitesse, Yolo maintient une haute précision de détection d'objets et fournit des résultats précis et fiables [13].
+ **Bonne généralisation** : YOLO se généralise bien à différentes scènes et situations en raison de son architecture robuste et à son entraînement sur différents ensembles de données [22].
+ **Open Source** : YOLO est une méthodologie open source qui offre accessibilité et flexibilité aux développeurs et aux chercheurs pour s'adapter à leurs besoins spécifiques [22].

2.4.3. L'Évolution du YOLO

Le tableau suivant résume l'évolution de YOLO:

Tableau 1. L'évolution de YOLO [24]

Version	Architecture	Optimisation	Base de données	Année de sortie
YOLOv1	24 couches convolutionnelles suivies de 2 couches entièrement connectées	SGD (Stochastic Gradient Descent)	PASCAL VOC (20 classes)	2016
YOLOv2	Darknet-19 avec 19 couches convolutionnelles + 5 couches max-pooling	Batch normalization, ancres multiples	PASCAL VOC, COCO (80 classes)	2016
YOLOv3	Darknet-53 avec 53 couches convolutionnelles	Binary Cross-Entropy, ancres multiples	COCO (80 classes)	2018
YOLOv4	CSPDarknet53 avec Cross Stage Partial connections	Optimisation MOS (Mosaic data augmentation, Self-Adversarial Training)	COCO (80 classes)	2020
YOLOv5	Diverses versions (Nano, Small, Medium, Large, XLarge) basées sur une architecture plus légère	Adam, optimisations spécifiques à l'inférence	COCO (80 classes)	2020
YOLOv6	Backbone CSPNet et PANet	Techniques de prunning et de quantization pour l'optimisation	COCO (80 classes)	2022
YOLOv7	Extended-ELAN pour une meilleure détection multi-échelle	Batch normalization, nouveaux modules d'apprentissage	COCO (80 classes)	2022
YOLOv8	CSPDarknet53 avec PANet, architecture modulaire	RAdam, MixUp, annealing scheduler	COCO (80 classes)	2023

+ **YOLOv1** a introduit l'idée de traiter la détection d'objets en une seule passe à travers le réseau, ce qui le rendait très rapide par rapport aux approches précédentes.
+ **YOLOv2** a amélioré la précision en utilisant Darknet-19 et en introduisant des ancres multiples et la normalisation par lots.

+ **YOLOv3** a encore augmenté la performance avec Darknet-53, permettant une meilleure extraction des caractéristiques grâce à ses 53 couches.
+ **YOLOv4** a introduit CSPDarknet53 et des techniques avancées d'optimisation comme la MOS et l'entraînement auto-adversarial, augmentant encore la précision et la vitesse.
+ **YOLOv5** est connu pour ses diverses versions adaptées à différents cas d'utilisation, allant des modèles très légers à des modèles plus complexes et précis.
+ **YOLOv6** a continué à améliorer les performances avec une meilleure optimisation pour l'inférence en temps réel.
+ **YOLOv7** a introduit de nouvelles techniques d'apprentissage et de normalisation pour améliorer la détection multi-échelle.
+ **YOLOv8** apporte des innovations en termes de structure modulaire, utilisant CSPDarknet53 avec PANet pour une meilleure fusion des caractéristiques et une formation plus efficace avec RAdam et MixUp (ar5iv) (MDPI) (LearnCV) (YOLOv8) .

Ces versions successives montrent une évolution constante pour améliorer à la fois la précision et la vitesse, répondant à des besoins toujours plus complexes dans le domaine de la détection d'objets en temps réel.

2.4.4. Les Étapes de l'algorithme YOLO

Avant de présenter les étapes nécessaires pour l'implémentation de l'algorithme YOLO, nous allons donner la définition de certains paramètres qui y sont utilisés.

Intersection sur Union (IoU) :

Cette étape évalue la précision des prédictions en mesurant le chevauchement entre les boîtes englobantes prédites et les boîtes englobantes réelles des objets, appelé IoU (Intersection over Union). L'IoU compare les boites englobantes prédites et détectées. Elle est calculée comme le rapport de l'aire de chevauchement à l'aire de l'union entre les deux boîtes [13]. Lorsque l'IoU dépasse un certain seuil, on déclare que l'objet est détecté.

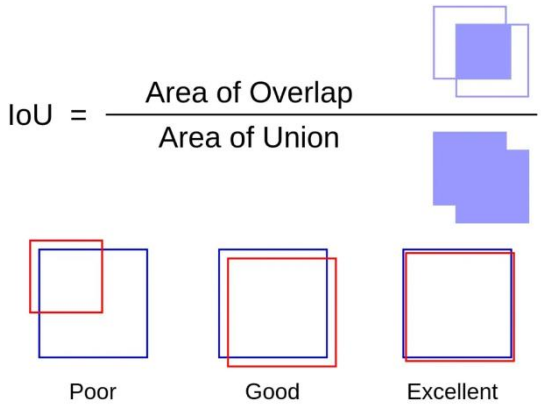


Figure 20. Définition de l'Intersection sur l'union (IoU)

Grille de Cellules:

L'idée est de diviser l'image d'entrée en une grille de cellules, généralement de taille S x S. Chaque cellule d'une grille est responsable de prédire les objets dont les centres se trouvent à son intérieur. Si le centre d'un objet appartient à une cellule de grille donnée, celle-ci prédît cet objet [22].

Boîte d'ancrage (Anchor Box) :

L'algorithme de YOLO utilise les boîtes d'ancrage pour séparer les objets lorsque plusieurs centres d'objets se trouvent dans la même cellule d'une grille. Elles offrent la possibilité de séparer et de localiser les objets de manière précise en leur attribuant des rectangles prédéfinis de dimensions et de formes précises [13]. Cela assure la bonne identification et la localisation de chaque objet, même lorsque les objets se superposent ou sont proches les uns des autres. Les boîtes de fixation jouent un rôle essentiel dans l'algorithme YOLO afin de détecter de manière précise les objets présents dans les images [22].

Comme illustré à la Figure 24, les paramètres définissant une boite englobante sont rangés dans un vecteur $Y = [p_c, b_x, b_y, b_h, b_w, c_1,\ldots,c_N]$, où :

- ✓ p_c correspond à la probabilité de la grille contenant l'objet
- ✓ b_x, b_y sont les coordonnées x et y du centre de la boite englobante par rapport à la cellule de la grille englobante.
- ✓ b_h, b_w correspondent à la hauteur et la largeur de la boite englobante par rapport à la cellule de la grille englobante.

- ✓ c_1, … c_N correspondent aux N classes que peut contenir une image (2 classes dans la figure ci-dessous : joueur et ballon).

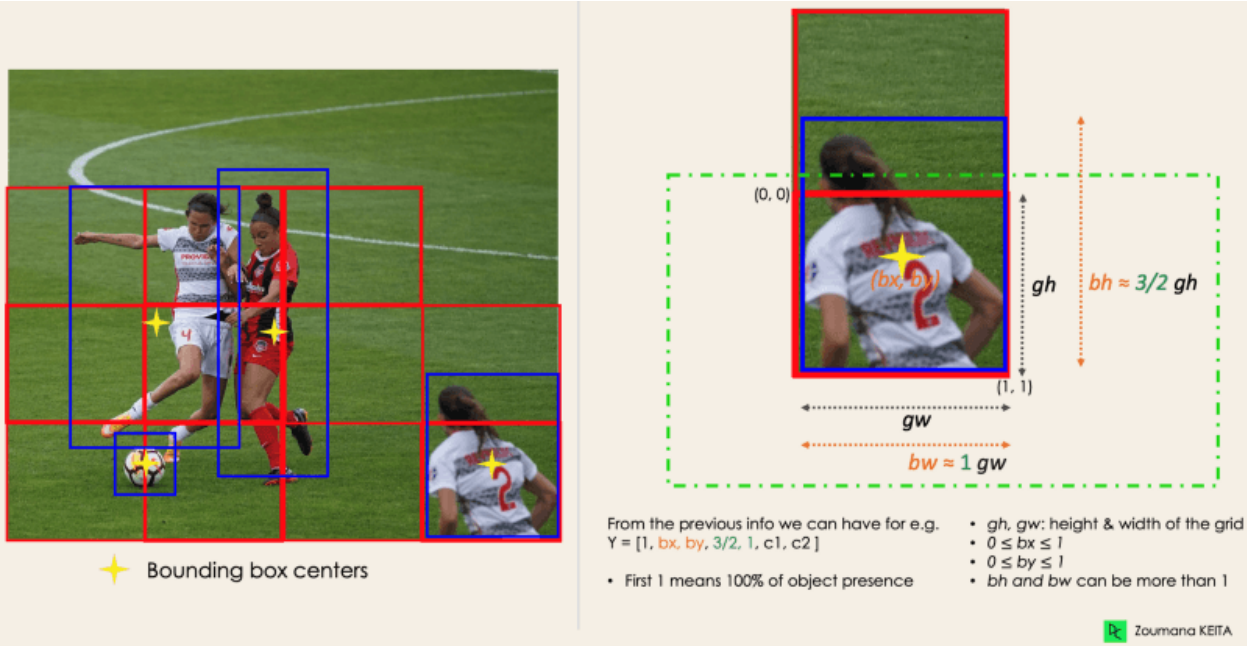


Figure 21. Boîte d'ancrage (Anchor Box) [22]

Suppression non maximale :

Cette étape est la dernière étape de l'algorithme de détection. Il s'agit d'un processus qui vise à supprimer les détections redondantes en éliminant les boites englobantes improbables [13]. La Figure 22 illustre ce processus.

La mise en œuvre de l'algorithme YOLO se déroule en 5 étapes [13] :

+ **Etape 01 :** Détection des boîtes englobantes les plus probables pour chaque objet détecté en attribuant une probabilité à chaque boîte pour chaque classe d'objet.
+ **Etape 02 :** Suppression des boîtes englobantes redondantes en conservant que la boîte avec la probabilité la plus élevée.
+ **Etape 03 :** Application d'un seuil de confiance pour éliminer les détections de faible probabilité, assurant ainsi que seules les détections fiables sont conservées.
+ **Etape 04 :** Identification des classes prédites ce qui permet d'associer chaque détection à une classe spécifique.

+ **Etape 05 :** Les résultats finaux de l'algorithme YOLO sont les détections qui restent après la suppression maximale, prêtes à être exploitées dans des applications de détection d'objets en temps réel.

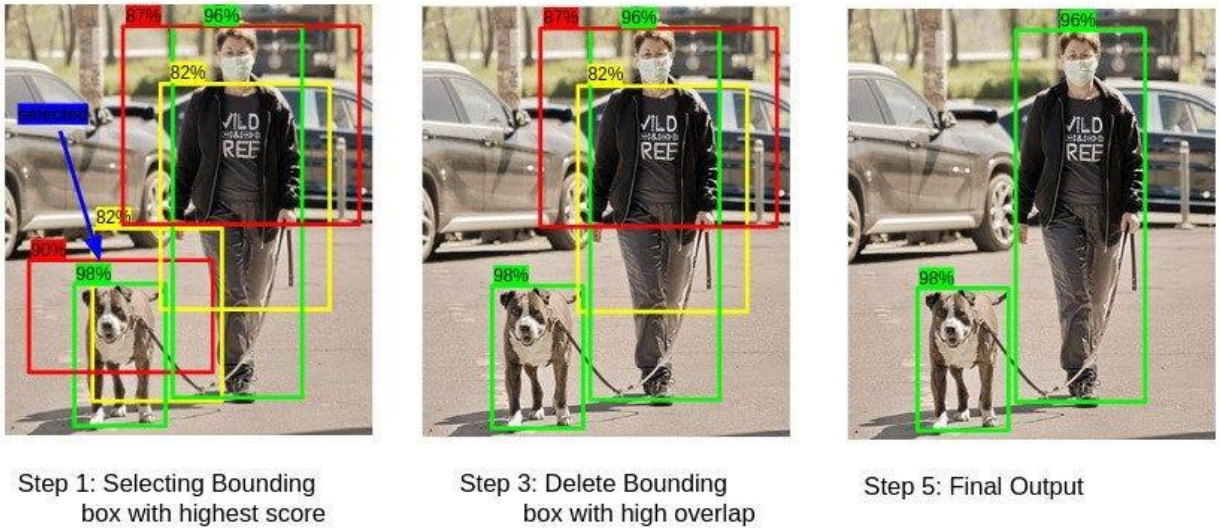


Figure 22. Suppression non maximale

2.4.5. Le Modèle YOLOv8

YOLOv8 est une technologie d'apprentissage profond de pointe qui a été spécialement conçue pour détecter des objets en temps réel dans les applications de vision informatique. YOLOv8 a bouleversé le domaine de la détection d'objets grâce à son architecture avancée et à ses algorithmes de pointe, offrant ainsi une détection précise et efficace d'objets dans des scénarios en temps réel [24].

YOLOv8 possède de nombreuses fonctionnalités performantes qui en font un choix remarquable pour les tâches de détection d'objets. Que l'on souhaite utiliser des modèles pré-entraînés ou concevoir des modèles sur mesure pour des types d'objets spécifiques, YOLOv8 propose une variété de fonctionnalités pour satisfaire les exigences.

Examinons en détail ces fonctionnalités essentielles [24] :

Tableau 2. Fonctionnalités du YOLOv8 [24]

Fonctionnalité	Description
Modèles pré-entraînés	Utiliser des modèles formés sur COCO pour une détection immédiate des objets
Modèles personnalisés	Créer des modèles spécialisés formés sur des types d'objets spécifiques
Préparation des données	Organiser et étiqueter des ensembles de données de formation pour une détection d'objets sur mesure
Prise en charge des applications Web	Créer des interfaces Web intuitives pour la détection d'objets en temps réel

2.4.5.1. Les différentes versions de YOLOv8

Pour illustrer l'efficacité et les différences entre les versions, voici un tableau comparatif des différentes versions de YOLOv8 :

Tableau 3. Les différentes versions de YOLOv8 [25].

Version	Architecture	Temps d'exécution (relatif)
YOLOv8-nano	Très légère, optimisée pour les appareils embarqués, quelques couches CNN	Très faible
YOLOv8-tiny	Architecture CNN compacte, balance entre taille et performance	Faible
YOLOv8-small	CNN avec quelques couches résiduelles, meilleure performance que tiny	Moyen
YOLOv8-medium	Plus de couches CNN et résiduelles, plus de capacité de modélisation	Modéré
YOLOv8-large	Profonde CNN avec de nombreuses couches résiduelles, meilleures performances	Élevé
YOLOv8-xlarge	Très profonde CNN, optimisée pour des environnements de haute performance	Très élevé

Détails de l'architecture :

+ **YOLOv8-nano** :
    - Architecture ultra-compacte
    - Quelques couches convolutionnelles basiques
    - Optimisé pour une latence très faible et des appareils à faible puissance
+ **YOLOv8-tiny** :
    - Compacte avec plusieurs couches convolutionnelles
    - Compromis entre taille et performance
    - Adapté pour les applications en temps réel sur des appareils mobiles
+ **YOLOv8-small** :
    - Inclut des couches résiduelles pour améliorer l'apprentissage
    - Meilleure précision que la version tiny
+ **YOLOv8-medium** :
    - Plus de couches et de filtres que la version small
    - Capacité de modélisation accrue, donc meilleure précision
+ **YOLOv8-large** :
    - Architecture plus profonde avec de nombreuses couches résiduelles
    - Conçu pour des environnements de calcul plus puissants
    - Précision significativement meilleure
+ **YOLOv8-xlarge** :
    - Très grande architecture CNN
    - Optimisé pour des systèmes de haute performance
    - Précision la plus élevée, mais demande de calcul également la plus élevée

2.5. Avantages et Inconvénients de YOLO par Rapport aux Autres Méthodes de Détection d'Objets

+ **Avantages :**

  Les caractéristiques suivantes démontrent les atouts significatifs du modèle YOLO dans le domaine de la détection d'objets en temps réel :

  - Traitement en temps réel : Yolo est réputé par sa rapidité permettant une détection en temps réel.
  - Détection d'objets à différentes échelles : Il est capable de détecter des objets de différentes tailles et formes.
  - Architecture simple et efficace : YOLO utilise une architecture simple et unifiée, facilitant la formation et le déploiement dans une variété de scénarios d'application, avec des performances compétitives.

+ **Inconvénients :**

  Malgré ses nombreux avantages, le modèle YOLO présente également certains inconvénients qu'il convient de prendre en compte :

  - Moindre précision par rapport à certaines méthodes.
  - Sensibilité aux déformations d'objets.
  - Difficulté de détection d'objets très petits.
  - Exigence de puissance de calcul.

3. Conclusion

En résumé, ce chapitre a présenté le modèle YOLO, qui est un modèle basé sur le deep learning, connu pour être à la pointe des algorithmes de détection des objets en temps réel. En particulier, la version récente YOLOv8 de cet algorithme possède des performances intéressantes en termes de précision et de rapidité, ce qui nous a poussé à l'utiliser pour la réalisation du prototype de feux tricolores intelligents, qui sera décrit dans le chapitre suivant.

Chapitre III : La conception du Prototype intégré de contrôle intelligent de feux tricolores

1. Introduction

Ce chapitre détaille la conception de notre système de feux tricolores intelligents. Nous décrivons d'abord le schéma général du système et ses composants principaux. Ensuite, nous présentons les plateformes et outils de programmation utilisés, tels que les langages de programmation et les bibliothèques (OpenCV, NumPy, skimage, SciPy), ainsi que Yolov8 pour la détection d'objets et SORT pour le suivi en temps réel via le filtre de Kalman.

Enfin, nous abordons le matériel employé, notamment le Raspberry Pi 4, la caméra web et les feux tricolores. Chaque section fournit une vue d'ensemble claire de la manière dont ces éléments collaborent pour optimiser la gestion du trafic urbain.

2. Conception du système

2.1. Schéma général

Le schéma général ci-dessous offre une vue d'ensemble du prototype de contrôle intelligent des feux tricolores. Il illustre le flux de données et les interactions entre les différents composants du système, depuis la capture des données jusqu'à la prise de décision. Chaque élément joue un rôle crucial, de la capture des images par la webcam à l'analyse des données par YOLOv8 pour la détection d'objets, en passant par le suivi des objets et la prise de décision. Ce schéma permet de visualiser clairement la manière dont les informations sont traitées et utilisées pour contrôler intelligemment les feux tricolores, contribuant ainsi à la gestion efficace du trafic urbain en temps réel.

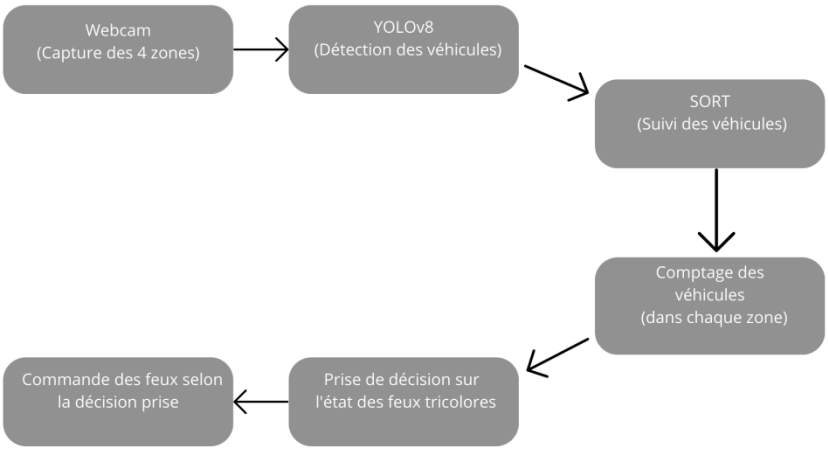


Figure 23. Schéma générale

Explications des Étapes :

1. **Webcam capture les 4 zones de l'intersection :** La webcam est positionnée pour couvrir l'ensemble de l'intersection, capturant en temps réel les images des quatre zones.
2. **YOLOv8 fait la détection des véhicules dans les 4 zones capturées par la webcam :** Les images capturées sont analysées par l'algorithme YOLOv8 pour détecter les véhicules présents dans chaque zone.
3. **Ces véhicules détectés vont être suivis par l'algorithme SORT :** Les véhicules détectés sont ensuite suivis à l'aide de l'algorithme de suivi d'objets SORT, ce qui permet de suivre les mouvements des véhicules dans les différentes zones.
4. **Après avoir fait la détection et le tracking, on fait le comptage des véhicules dans chaque zone pour prendre une décision sur l'état des feux tricolores dans l'intersection :** Le nombre de véhicules dans chaque zone est compté, et ces données sont utilisées pour déterminer l'état optimal des feux tricolores (par exemple, prolonger le feu vert pour une zone avec un trafic dense).
5. **Commande des feux selon la décision prise :** L Le microprocesseur, selon la décision basée sur le comptage des véhicules, ajustera les feux tricolores en conséquence pour optimiser le flux de trafic.

2.2. Plateformes et outils de programmation utilisés

Dans cette section, nous examinons les plateformes, les outils de programmation ainsi que le langage utilisé dans la conception du prototype de contrôle intelligent de feux tricolores. Ces éléments sont cruciaux pour la capture, le traitement et l'analyse des données en temps réel. Nous abordons également le langage de programmation Python et les bibliothèques OpenCV, NumPy, Scikit-image et SciPy associées qui ont été utilisées pour créer et optimiser les fonctionnalités du système.

2.2.1. Langage de programmation et bibliothèques utilisées

Python [28] **:**

Python, un langage de programmation de haut niveau, polyvalent et extrêmement populaire, a été le langage principal utilisé dans l'implémentation du prototype. Référencé dans de nombreuses applications, le langage de programmation Python (en particulier la dernière version, Python 3) est largement utilisé dans le développement Web, les applications d'apprentissage automatique et les technologies de pointe de l'industrie logicielle. Il est apprécié tant par les débutants que par les programmeurs expérimentés dans d'autres langages tels que C++ et Java.

Figure 24. Python

Les étapes d'implémentation sont réalisées à l'aide de diverses bibliographies telles que : opencv ,NumPy, Scikit-image, SciPy

Opencv [29]**:**

OpenCV est une bibliothèque open source largement utilisée pour le traitement d'images et la vision par ordinateur. Elle offre une gamme étendue de fonctionnalités avancées, notamment la détection d'objets, la reconnaissance faciale, le suivi de mouvement, la calibration de caméra, etc. Son architecture modulaire et sa grande compatibilité en font un outil de choix pour les projets de vision par ordinateur.

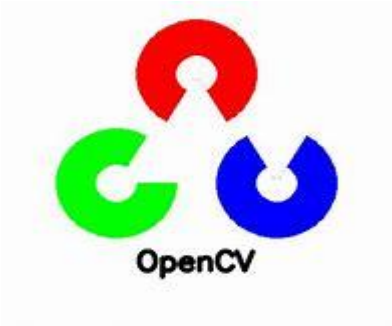


Figure 25. Opencv

NumPy [30]**:**

NumPy est une bibliothèque fondamentale pour le calcul numérique en Python. Elle fournit des structures de données puissantes, telles que des tableaux multidimensionnels, ainsi qu'une large collection d'opérations mathématiques avancées et de fonctions pour la manipulation de données. Grâce à sa performance élevée, NumPy est largement utilisée dans les domaines scientifiques et d'ingénierie.

Figure 26. NumPy

Sckit-image [31]**:**

Scikit-image est une bibliothèque Python dédiée au traitement d'images. Elle propose une multitude de fonctionnalités pour la segmentation d'images, la détection de contours, la reconnaissance de formes, la restauration d'images, et bien plus encore. En offrant des outils puissants et conviviaux, Scikit-image est un choix populaire pour les projets de traitement d'images en Python.

Figure 27. Sckit-image

Scipy [32]**:**

SciPy est une bibliothèque Python spécialisée dans les calculs scientifiques et techniques. Elle fournit une large gamme d'outils et d'algorithmes pour l'optimisation, l'algèbre linéaire, l'intégration numérique, l'analyse statistique, etc. Avec son écosystème riche et mature, SciPy est largement utilisée dans la recherche scientifique, l'ingénierie et d'autres domaines.

Figure 28. SciPy

RPi. GPIO :

La bibliothèque RPi.GPIO est utilisée pour contrôler les broches GPIO du Raspberry Pi. Elle permet d'interagir avec des composants électroniques, tels que les LED et les capteurs, via des scripts Python.

2.2.2. YOLOv8 pour la détection d'objets

Dans ce projet, nous exploitons des détecteurs d'objets tels que YOLOv8, spécifiquement formés pour la détection d'une variété d'objets. Le processus d'entraînement implique l'utilisation d'un ensemble d'images avec leurs annotations correspondantes pour ajuster le modèle et lui permettre de reconnaître efficacement les objets. Une fois l'entraînement achevé, nous obtenons un fichier modèle contenant les connaissances acquises par le modèle YOLOv8. Ce modèle pré-entraîné est capable de détecter divers types d'objets, notamment des personnes, des voitures, des vélos, des chiens, des chats, des avions, des bateaux, et bien d'autres. [24]

Dans notre programme, nous effectuons quatre étapes simples :

1. **Initialisation du modèle** : Nous importons le modèle pré-entraîné YOLOv8 à l'aide de la classe YOLO fournie par la bibliothèque Ultralytics dans notre programme.
2. **Détection d'objets** : Pour chaque vidéo capturée par la caméra, le modèle YOLOv8 est utilisé pour détecter les objets présents dans chaque frame.
3. **Traitement des résultats :** Les résultats de la détection d'objets sont ensuite traités pour extraire les coordonnées des boîtes englobantes des objets détectés, ainsi que leur classe et la confiance associée à chaque prédiction. Seuls les objets de certaines classes (tels que les voitures, les camions, les bus, etc.) dont la confiance dépasse un seuil défini sont conservés pour une utilisation ultérieure.
4. **Utilisation des informations obtenues :** Dans ce programme, les informations obtenues à partir de la détection et du suivi des objets sont utilisées de manière stratégique pour le contrôle du trafic.

Figure 29. Yolov8

2.2.3. Sort (poursuite par le filtre de Kalman) [33]

L'algorithme SORT (Simple Online and Realtime Tracking) est une méthode de suivi des objets en temps réel largement utilisée dans le domaine de la vision par ordinateur. Il combine plusieurs concepts et techniques, dont le filtre de Kalman, pour estimer la trajectoire des objets dans une séquence vidéo.[33]

2.2.3.1. Filtre de Kalman [33]

+ Le filtre de Kalman est un algorithme de récursion qui est utilisé pour estimer l'état d'un système dynamique à partir d'une série de mesures bruitées.
+ Dans le contexte du suivi d'objets, le filtre de Kalman est utilisé pour prédire la position future d'un objet en mouvement, en tenant compte à la fois de sa position actuelle et de sa vitesse.
+ Le filtre de Kalman estime également l'incertitude associée à chaque prédiction, ce qui permet de pondérer les nouvelles observations en fonction de leur fiabilité.

2.2.3.2. SORT (Simple Online and Realtime Tracking)[33]

+ L'algorithme SORT utilise le filtre de Kalman pour prédire les emplacements futurs des objets dans une séquence vidéo.
+ Il associe ensuite les détections actuelles (cadres) aux objets prédits en utilisant une mesure de similarité basée sur l'intersection sur l'union (IoU) [13]entre les boîtes englobantes des détections et des prédictions.
+ Les détections qui ne sont pas associées à des objets prédits sont considérées comme de nouveaux objets et des nouveaux trackers (filtres de Kalman) sont initialisés pour les suivre.
+ L'algorithme gère également les cas où un objet disparaît temporairement de la scène (par exemple, lorsqu'il est temporairement caché par un autre objet) en continuant à prédire sa trajectoire pendant un certain nombre de frames sans détection associée.

En résumé, l'algorithme SORT combine le filtre de Kalman pour la prédiction de trajectoire avec des techniques d'association de détections pour suivre efficacement les objets dans une séquence vidéo en temps réel.

2.3. Matériel utilisé pour réaliser le systeme

2.3.1. Raspberry Pi 4 [34]

Le Raspberry Pi 4 est un micro-ordinateur monocarte développé par la Fondation Raspberry Pi [34]. Il offre des fonctionnalités avancées et des performances accrues par rapport aux modèles précédents, ce qui en fait un choix idéal pour notre application de suivi d'objets en temps réel. Voici quelques caractéristiques clés du Raspberry Pi 4 qui le rendent adapté à notre projet :

Performances améliorées : Le Raspberry Pi 4 est équipé d'un processeur quad-core ARM Cortex-A72 cadencé jusqu'à 1,5 GHz, offrant des performances nettement supérieures à celles des modèles précédents. Cette puissance de traitement accrue est essentielle pour exécuter efficacement des algorithmes de suivi d'objets en temps réel.

Mémoire RAM étendue : Le Raspberry Pi 4 est disponible avec différentes configurations de mémoire RAM, allant de 2 Go à 8 Go. Une mémoire RAM plus importante permet de manipuler des données plus volumineuses et d'exécuter simultanément plusieurs processus, ce qui est nécessaire pour un suivi d'objets fluide et réactif.

Connectivité améliorée : Le Raspberry Pi 4 est doté de multiples ports USB 3.0, de ports Gigabit Ethernet et de connexions sans fil Wi-Fi et Bluetooth intégrées. Cette connectivité étendue nous permet de connecter facilement plusieurs webcams pour la capture de flux vidéo et de communiquer avec d'autres dispositifs externes, tels que les feux tricolores à LEDs.

Support de sortie vidéo 4K : Le Raspberry Pi 4 prend en charge la sortie vidéo 4K via son port HDMI, offrant une qualité d'affichage exceptionnelle. Bien que notre application de suivi d'objets n'exploite pas directement cette fonctionnalité, elle garantit une qualité d'image élevée pour la visualisation des résultats du suivi sur un écran externe.

Polyvalence et faible coût : Le Raspberry Pi 4 est un dispositif polyvalent et abordable, ce qui en fait une solution attrayante pour une large gamme de projets informatiques. Sa faible consommation d'énergie le rend également adapté à des applications embarquées ou mobiles, telles que notre système de suivi d'objets installé dans un véhicule.

En raison de ses performances améliorées, de sa connectivité étendue et de sa polyvalence, nous avons choisi le Raspberry Pi 4 comme plateforme de base pour notre projet de suivi d'objets en temps réel. Son architecture ouverte et sa large communauté de développeurs offrent également un support et des ressources précieux pour le développement et la mise en œuvre de notre système.

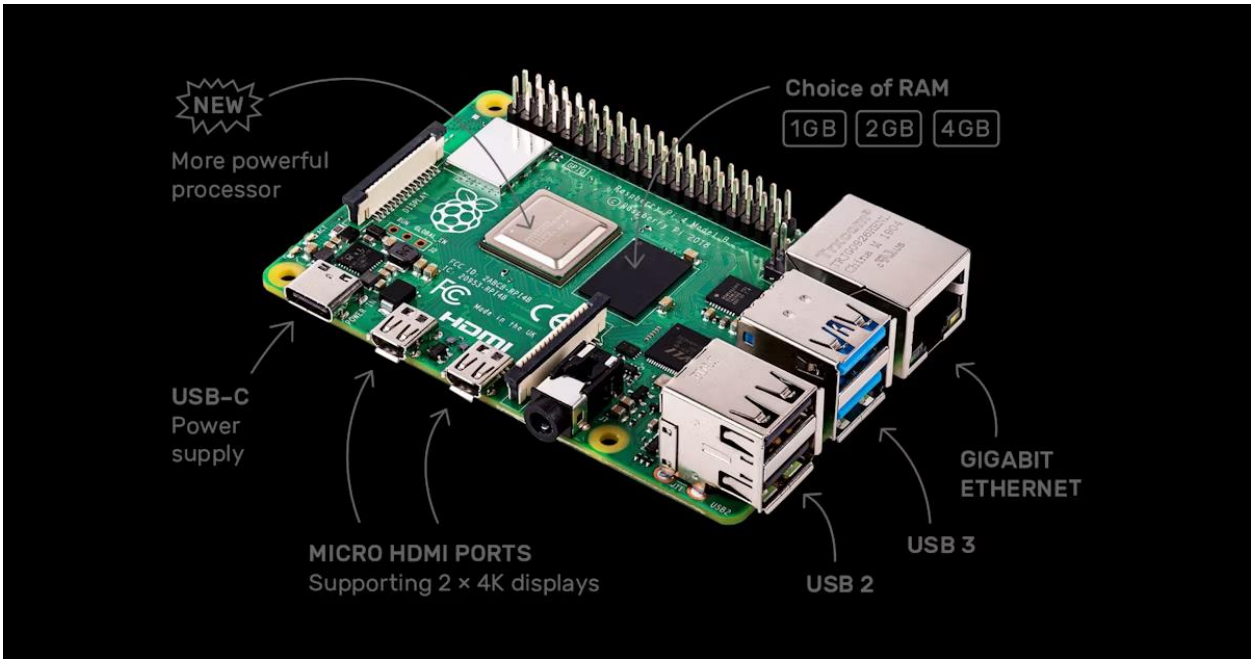


Figure 30. Raspberry Pi 4

2.3.2. Caméra web

Pour capturer les flux vidéo nécessaires au suivi des objets, nous avons utilisé quatre webcams USB standard du même type, connectées au Raspberry Pi 4 [34]via ses ports USB 3.0. Voici les spécifications techniques de la webcam utilisée présente dans la figure 31 :

+ **Logitech C920 HD Pro Webcam** :
    - Résolution : Full HD 1080p (1920 x 1080 pixels)
    - Fréquence d'images : Jusqu'à 30 images par seconde (fps)
    - Champ de vision : 78 degrés
    - Microphone intégré pour l'audio
    - Connexion : USB 2.0/3.0

Figure 31. Logitech C920 HD Pro Webcam

Ces webcams offrent des résolutions allant de HD 720p à Full HD 1080p, ce qui garantit une qualité vidéo adéquate pour la détection et le suivi des objets. La fréquence d'images maximale de 30 fps permet d'obtenir des vidéos fluides et réactives, ce qui est crucial pour le suivi en temps réel.

Pour connecter la webcam au Raspberry Pi 4[34], nous avons simplement branché la webcam sur un port USB disponible du Raspberry Pi. Une fois connectées, le Raspberry Pi est capable de détecter automatiquement la webcam et de l'utiliser comme périphériques vidéo pour la capture de flux en temps réel.

En utilisant une seule webcam, notre système capture des vidéos à partir d'une perspective unique. Cela simplifie l'installation et réduit les coûts tout en maintenant une précision suffisante pour la détection et le suivi des objets. Cette approche permet de surveiller efficacement le mouvement des objets dans l'environnement avec des ressources limitées.

2.3.3. Feux tricolores

Dans la construction des feux tricolores à l'aide de LEDs et de câbles, nous avons utilisé les composants suivants pour chaque feu tricolore :

+ **LEDs** : Nous avons utilisé des LEDs de trois couleurs différentes pour représenter les feux tricolores. Habituellement, ces LEDs sont de couleur rouge, jaune et verte, respectivement, pour représenter les feux d'arrêt, de précaution et de départ.

+ **Résistances** : Pour chaque LED, nous avons ajouté une résistance appropriée pour limiter le courant et protéger la LED contre une surintensité. La valeur de la résistance dépend du type de LED et de la tension d'alimentation.
+ **Fils de connexion** : Nous avons utilisé des fils de connexion pour relier les LEDs et les résistances selon le schéma de câblage requis.
+ **Plaque de prototypage (breadboard)** : Une plaque de prototypage a été utilisée pour organiser les composants et faciliter les connexions entre eux.

Pour réaliser quatre feux tricolores, le schéma de câblage pour connecter les LEDs au Raspberry Pi variera en fonction du nombre de LEDs utilisées et de la disposition des feux tricolores. Habituellement, chaque LED est associée à une broche GPIO (General Purpose Input/Output) distincte du Raspberry Pi. Voici un exemple simplifié de schéma de câblage pour un feu tricolore unique :

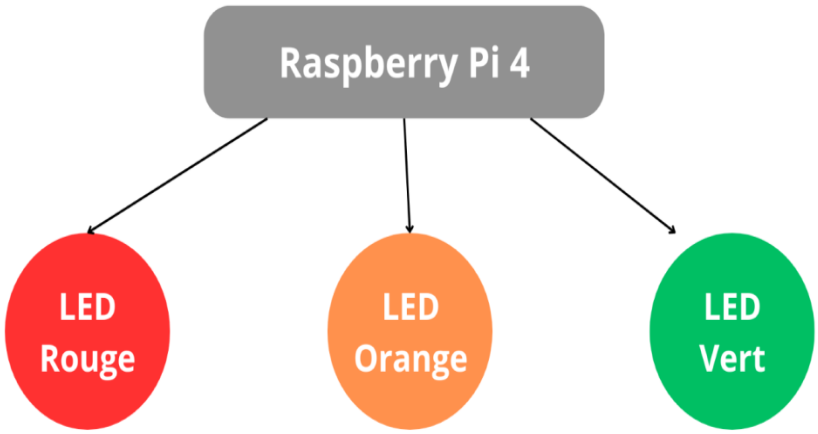


Figure 32. Un schéma de câblage simplifié

3. Conclusion

Dans ce chapitre, nous avons détaillé la conception de notre système de feux tricolores intelligents. Nous avons décrit le schéma général du système, les plateformes et outils de programmation utilisés (OpenCV, NumPy, skimage, SciPy, Yolov8, SORT et le filtre de Kalman), ainsi que le matériel (Raspberry Pi 4, webcam unique et feux tricolores). Cette conception nous a permis de poser des bases solides pour le développement de notre solution.

Chapitre IV : Implémentation et test du système conçu

1. Introduction

Nous allons maintenant passer à l'implémentation pratique et les tests de validation du système conçu.

Ce chapitre abordera la réalisation du prototype, la mise en œuvre de la détection par Yolov8 et du suivi par l'algorithme SORT, la mise en place des règles de commande des feux tricolores, ainsi que l'implémentation sur la Raspberry Pi 4. Enfin, nous effectuerons des tests de validation pour évaluer les performances et l'efficacité de notre système.

2. Réalisation du prototype

Dans cette section, nous décrivons la réalisation pratique de notre prototype de contrôle intelligent de feux tricolores, basé sur la détection et le suivi d'objets.

2.1. Maquette de l'intersection

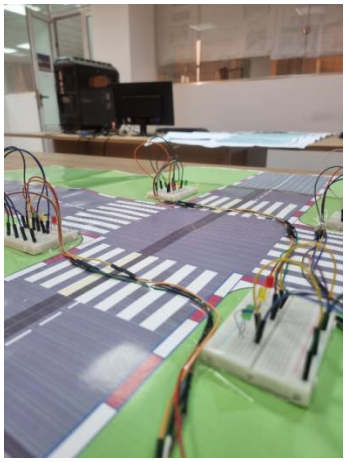
La maquette de l'intersection, comme le montre la Figure 33, a été conçue pour reproduire fidèlement les conditions d'une intersection routière réelle. Elle comprend :

- Quatre voies de circulation (deux dans chaque direction)
- Des trottoirs et des zones de passage piéton
- Des feux tricolores positionnés aux emplacements appropriés de l'intersection pour réguler la circulation

Figure 33. Maquette de l'intersection

2.2. Placement des feux tricolores

Les feux tricolores sont placés conformément aux normes de signalisation routière en vigueur. À chaque voie à l'approche de l'intersection, un ensemble de feux tricolores est installé pour contrôler la circulation. Les feux sont disposés de manière à être clairement visibles par les conducteurs et les piétons, comme le montre la Figure 34.



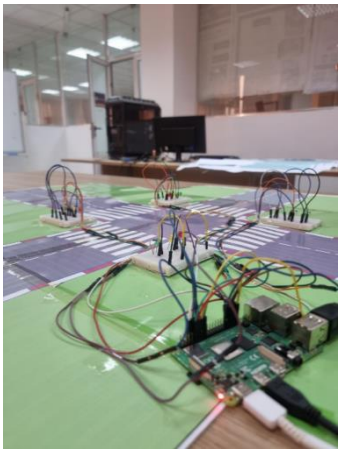


Figure 34. Placement des feux tricolores à l'intersection.

2.3.Emplacement de la caméra

Une caméra est montée en hauteur, offrant une vue panoramique de l'intersection. Son emplacement a été soigneusement étudié pour maximiser la couverture de la zone et permettre une détection précise des véhicules et des piétons. La caméra est positionnée de manière à avoir une vue dégagée des feux tricolores et des différentes voies de circulation, comme le montre la Figure 35.

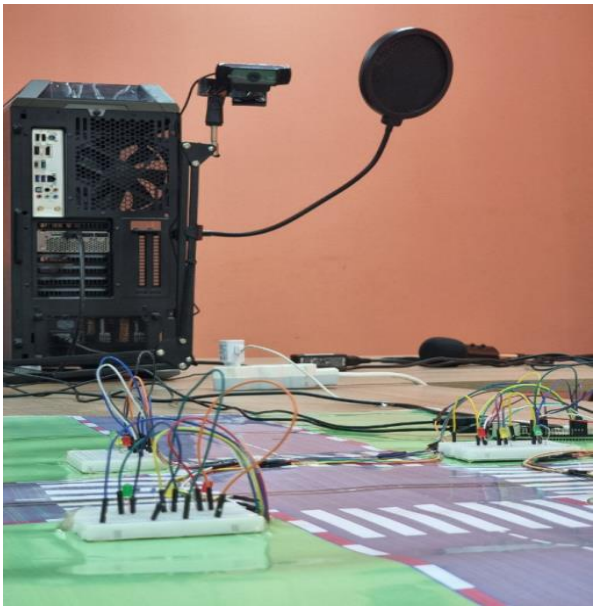


Figure 35. Placement de la caméra à l'intersection

La Figure 36 montre la maquette de l'intersection avec le placement des feux tricolores et de la caméra.

Figure 36. Maquette de l'intersection avec placement des feux tricolores et de la caméra.

3. Implémentation de la détection par Yolo et de la poursuite par l'algorithme Sort

Le système de gestion des feux tricolores repose sur l'utilisation d'un modèle YOLOv8n, spécialement entraîné pour reconnaître une variété d'objets, notamment les véhicules tels que les voitures, les camions, les motos, et bien d'autres. En parallèle, l'algorithme SORT est employé pour suivre ces objets détectés.

Tout d'abord, nous importons les packages nécessaires pour que notre modèle puisse fonctionner :

```
import numpy as np
import cv2
from ultralytics import YOLO
import time
import RPi.GPIO as GPIO
from sort import *
```

Figure 37. Les packages utilisés

Ensuite, nous chargeons notre modèle pour cette tâche spécifique en utilisant la ligne suivante :

```
model = YOLO("../Yolo-Weights/yolov8n.pt")
```

Figure 38. Code pour charger le yolov8

Une fois le modèle chargé avec succès, nous sommes prêts à l'utiliser pour détecter les objets dans nos flux vidéo en temps réel. Chaque image de la vidéo capturée par la webcam est passée au modèle YOLOv8n pour effectuer dessus la détection d'objets. Voici les étapes processus de détection en détail :

1. Capturer la vidéo à partir de la webcam.
2. Prétraiter la vidéo si nécessaire (par exemple, redimensionner, normaliser les pixels).
3. Faire une boucle pour traiter chaque image de la vidéo capturée.
4. Passer l'image prétraitée au modèle YOLOv8n pour effectuer la détection d'objets.
5. Recevoir les prédictions du modèle, y compris les boîtes englobantes des objets détectés et leurs classes.
6. Traiter les prédictions pour afficher les objets détectés sur l'image.

Voici un exemple de code illustrant ce processus :

```python
while True:
    success, frame = cap.read()  # Capturer une image à partir de la webcam
    if not success:
        break

    # Prétraiter l'image si nécessaire
    processed_frame = preprocess(frame)

    # Passer l'image prétraitée au modèle YOLOv8 pour la détection d'objets
    results = model(processed_frame, stream=True)

    # Traiter les prédictions pour afficher les objets détectés sur l'image
    for result in results:
        boxes = result.boxes
        confidences = result.conf
        classes = result.cls

        for box, conf, cls in zip(boxes, confidences, classes):
            x1, y1, x2, y2 = box.xyxy[0]
            label = classNames[int(cls)]
            confidence = round(float(conf), 2)
            cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
            cv2.putText(frame, f"{label}: {confidence}", (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

    cv2.imshow("Object Detection", frame)  # Afficher l'image avec les objets détectés

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Figure 39. Code du processus Yolov8n.

Ce code capture en boucle des images à partir de la webcam, détecte les objets dans chaque image à l'aide du modèle YOLOv8, et affiche les objets détectés avec leurs classes et confiances sur l'image.

Après avoir détecté les objets dans chaque image à l'aide de YOLOv8, nous utilisons l'algorithme SORT pour suivre ces objets dans le temps. Voici comment nous procédons :

1. Nous initialisons l'algorithme SORT avec les paramètres appropriés, tels que la durée maximale de suivi pour un objet, le seuil de chevauchement pour fusionner les boîtes englobantes, etc.
2. Pour chaque image traitée par YOLOv8, nous utilisons les boîtes englobantes des objets détectés pour mettre à jour l'état de suivi de l'algorithme SORT.
3. L'algorithme SORT associe les boîtes englobantes actuelles aux objets suivis précédemment en utilisant des techniques de suivi par correspondance, telles que la minimisation de la distance euclidienne.

4. Les objets suivis sont mis à jour en fonction des nouvelles détections, et les objets non suivis sont marqués comme perdus après un certain nombre de trames sans détection.
5. Les boîtes englobantes des objets suivis sont utilisées pour afficher les objets suivis sur l'image avec leur identifiant unique comme le montre la figure suivante.

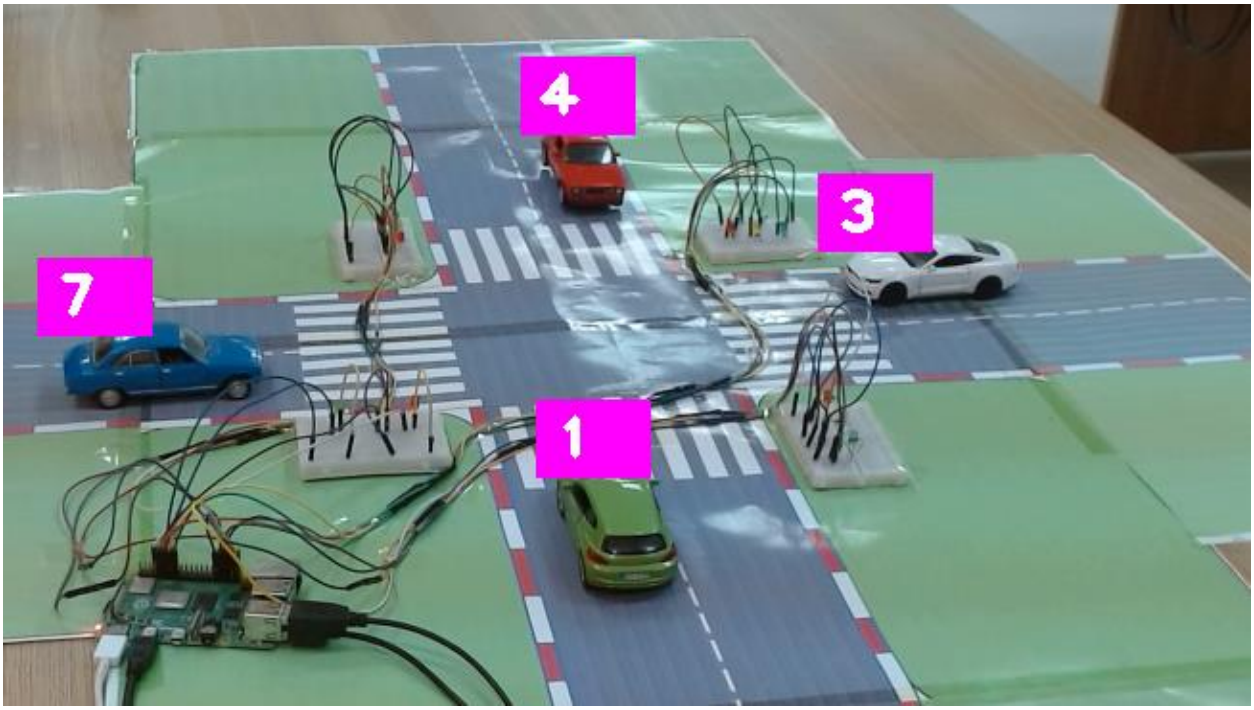


Figure 40. Identification des voitures détectées

Une la détection par YOLO et de la poursuite par l'algorithme SORT effectuées, nous passons à l'étape cruciale de l'implémentation des règles de commande des feux tricolores. Cette étape est essentielle pour garantir un contrôle efficace du trafic au niveau de l'intersection.

4. Implémentation des règles de commande des feux tricolores

Dans cette section, nous décrivons l'algorithme et les règles que nous avons utilisés pour un contrôle intelligent des feux tricolores. Le but est d'optimiser la fluidité du trafic et de garantir la sécurité des véhicules à l'intersection. Pour ce faire, nous avons mis en place une logique de prise de décision dynamique en temps réel, basée sur les résultats de de détection à partir des images capturées par la caméra.

4.1. Logique de base des feux tricolores

L'allumage des feux tricolores suit une séquence standard : vert, orange, puis rouge. Cette séquence permet de gérer efficacement le flux de véhicules à l'intersection, lorsque les durées des feux sont ajustées dynamiquement en fonction des conditions de trafic détectées. Le feu vert permet aux véhicules de passer, sa durée est ajustée en fonction du nombre de véhicules détectés. Le feu orange avertit les conducteurs que le feu va passer au rouge, avec une durée minimale de 2 secondes. Enfin, le feu rouge arrête les véhicules pour permettre aux véhicules circulant dans la direction perpendiculaire de passer, avec une minimale de 2 secondes.

4.2. Algorithme de contrôle des feux

L'algorithme de contrôle des feux tricolores suit les étapes suivantes :

1. **Initialisation** :
   - Configuration des broches GPIO pour contrôler les LEDs des feux tricolores.
   - Définition des zones d'intérêt pour la détection de véhicules.
2. **Cycle de détection et de décision** :

Étape 1 : Détection :

- Capture d'images et détection des véhicules dans les zones d'intérêt.
- Comptage des véhicules dans chaque zone.

Étape 2 : Calcul des durées :

- Comparaison du nombre de véhicules dans les zones 1 et 2 à celui dans les zones 3 et 4.
- Ajustement des durées des feux verts en fonction des résultats de la détection.

Étape 3 : Mise à jour des feux :

- Basculement des feux tricolores en appliquant les règles de commande.
- Fixation de la durée de maintien de chaque phase pour chaque feu.

3. **Boucle continue** :
    - Répétition du cycle de détection et de décision à intervalles réguliers pour une adaptation en temps réel. La période de cette répétition a été arbitrairement fixée à 2 secondes, mais dans une situation réelle il faut la fixer en fonction des conditions propres à chaque intersection.

4.3. Règles de commande des feux tricolores

Après la détection des véhicules dans chaque zone d'intérêt, nous calculons toutes les 20 secondes la somme des véhicules détectés dans les zones 1 et 2 et la somme des véhicules détectés dans les zones 3 et 4 et nous appliquons les règles suivantes pour décider de l'état des feux tricolores :

1. **Conditions normales** :

Si le total des véhicules dans les zones 3 et 4 est supérieur à celui des zones 1 et 2, les feux des intersections 3 et 4 passent au vert, et ceux des intersections 1 et 2 passent au rouge. Le compteur de feux verts pour les intersections 3 et 4 est incrémenté.

Inversement, si le total des véhicules dans les zones 3 et 4 est inférieur à celui des zones 1 et 2, les feux des intersections 1 et 2 passent au vert, et ceux des intersections 3 et 4 passent au rouge. Le compteur de feux verts pour les intersections 1 et 2 est incrémenté.

2. **Conditions spéciales** :
    - Si aucun véhicule n'est détecté dans toutes les zones, tous les feux passent au rouge.
    - Si les totaux de véhicules dans les deux groupes sont très élevés (plus de 30) ou égaux, un choix aléatoire est fait pour décider quelle direction passe au vert.

4. **Réinitialisation des compteurs** :

Après chaque décision, les compteurs de véhicules dans chaque zone sont réinitialisés à 0.

5. **Cycle des feux orange** :

Lors du changement des feux de vert vers rouge, une phase de feu orange de 2 secondes est activée pour avertir les conducteurs avant de passer au rouge.

6. **Limitation de la durée des feux verts** :

Si le nombre de véhicules dans les deux zones parallèles dépasse celui des deux autres zones, les deux zones parallèles avec le plus de véhicules auront le feu vert. Lors de la deuxième détection, si le système confirme que les deux premières zones parallèles comptent toujours plus de véhicules, les feux restent verts pour ces zones. Cependant, à la troisième détection, même si les deux autres zones ont moins de véhicules, elles recevront le feu vert. Si ces deux dernières zones sont vides, les feux restent verts pour les deux premières zones.

5. Implémentation sur la Raspberry Pi 4

Dans cette section, nous décrirons comment le système a été déployé sur la Raspberry Pi 4, en détaillant les étapes d'installation, la configuration des composants matériels, ainsi que les problèmes rencontrés et les solutions apportées.

Configuration de la Raspberry Pi 4

1.1. Installation du système d'exploitation

Pour installer le système d'exploitation sur la Raspberry Pi 4, nous avons utilisé Raspberry Pi Imager. Tout d'abord, nous avons téléchargé Raspberry Pi Imager depuis le site officiel et l'avons installé sur notre ordinateur. Ensuite, nous avons inséré une carte SD dans l'ordinateur, ouvert Raspberry Pi Imager, sélectionné "Choose OS" et choisi "Raspberry Pi OS (32-bit)". Nous avons sélectionné la carte SD insérée en cliquant sur "Choose SD Card" puis sur "Write" pour écrire l'image sur la carte SD. Une fois l'image écrite, nous avons inséré la carte SD dans la Raspberry Pi 4 et l'avons allumée. Enfin, nous avons suivi les instructions à l'écran pour la configuration initiale du système, y compris la connexion au réseau Wi-Fi et la mise à jour du système.

1.2. Installation des bibliothèques et dépendances

Une fois le système d'exploitation installé, nous avons procédé à l'installation des bibliothèques et dépendances nécessaires pour notre projet. Nous avons d'abord mis à jour le système en utilisant les commandes “**sudo apt update”** et “**sudo apt upgrade”**. Ensuite, nous avons installé les bibliothèques Python essentielles, notamment “**numpy”**,” **opencv-python"** et

"**RPi.GPIO"**, en utilisant "**pip3"** après avoir installé "**python3-pip"**. Pour utiliser la webcam, nous avons installé les pilotes nécessaires pour les webcams.

1.3. Tests de mise en route

Pour s'assurer du bon fonctionnement de la Raspberry Pi 4, nous avons réalisé une série de tests. Cela comprenait des tests de connexion de la webcam et des feux tricolores aux ports appropriés, et des tests d'interaction avec les feux tricolores réels pour vérifier la synchronisation et la réactivité du système. Nous avons également surveillé les performances du système pendant ces tests en mesurant la consommation de ressources et en évaluant la stabilité globale.

1.4 Résultats et Observations :

Lors du test du système, nous avons observé que la performance de la Raspberry Pi 4 était assez limitée. En particulier, le système était très lent et les programmes prenaient beaucoup de temps pour s'exécuter, principalement en raison des exigences élevées du traitement d'image et de la détection d'objets en temps réel. Le processeur de la Raspberry Pi 4 avait du mal à gérer les tâches intensives de traitement d'image nécessaires pour notre projet, ce qui entraînait des latences significatives et une réactivité réduite du système global. Ces limitations de performance ont révélé la nécessité d'optimisations supplémentaires ou d'un matériel plus puissant pour atteindre des performances acceptables.

Face aux limitations de performance rencontrées avec la Raspberry Pi 4, nous avons pris la décision d'éliminer le suivi d'objets de notre système. Cependant, même après avoir supprimé le tracking, le problème persistait. La Raspberry Pi 4 continuait à présenter des ralentissements significatifs et une exécution lente, compromettant ainsi la fonctionnalité globale du système. Ces résultats ont souligné les défis inhérents à la réalisation d'un système de traitement d'image et de détection d'objets sur un matériel aux capacités limitées comme la Raspberry Pi 4.

Pour pallier les limitations de ressources de la Raspberry Pi 4, nous avons opté pour une nouvelle approche : diviser notre système en une partie serveur et une partie client. La partie serveur, hébergée sur un serveur disposant de ressources élevées, prend en charge les tâches intensives de traitement d'image et de détection d'objets.

La Raspberry Pi 4, désormais en tant que client, exécute un code léger chargé de d'envoyer les images captées par les caméras au serveur distant, qui les traite et renvoie les instructions de contrôle des feux tricolores.

Cette nouvelle architecture serveur-client nous a permis de déployer un système fonctionnel, même si notre objectif initial d'un prototype autonome a été temporairement déplacé. Nous pensons que cet objectif pourrait être atteint en utilisant une carte plus puissante à l'image de la Raspberry Pi 5, jumelée à une carte graphique.

6. Tests de validation

Après avoir implémenté notre système de feux tricolores intelligents, il est crucial de vérifier son bon fonctionnement par des tests rigoureux. Cette section décrit les procédures de tests que nous avons suivies pour valider les performances de notre système.

6.1. Objectifs des Tests

Les principaux objectifs des tests de validation sont :

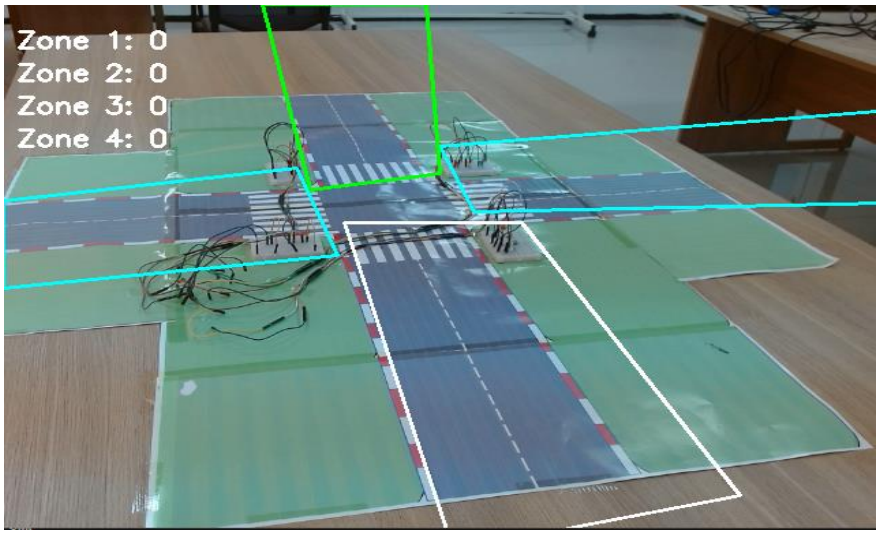
- Vérifier la précision de la détection et du suivi des objets (véhicules et piétons) par YOLOv8 et l'algorithme SORT.
- Évaluer la réactivité et l'efficacité des règles de commande des feux tricolores dans différentes conditions de trafic.
- Mesurer les performances du système implémenté sur la Raspberry Pi 4 et la l'architecture client-serveur proposée.

6.2. Tests effectués

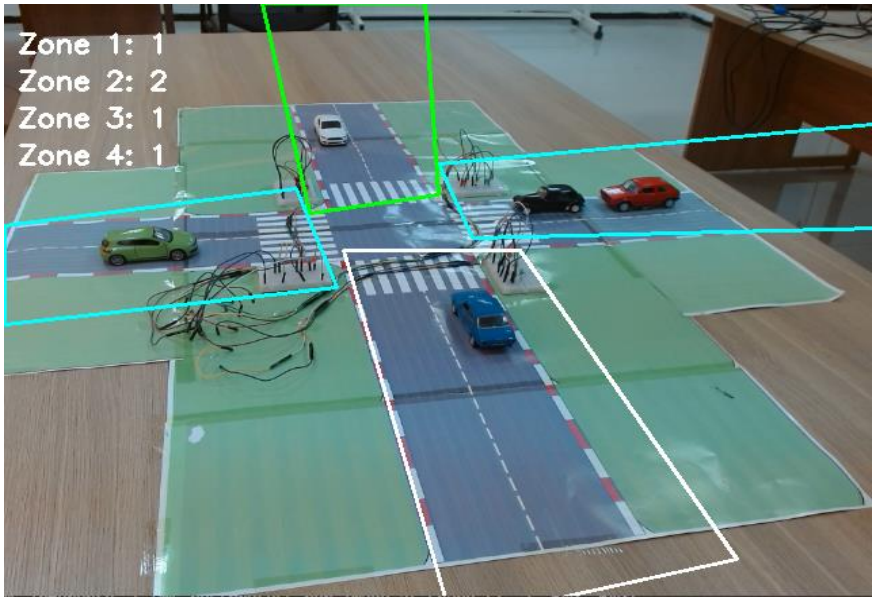
6.2.1. Test de Détection et Suivi d'Objets

- **Scénarios de Test** : Nous avons utilisé plusieurs scénarios de trafic simulés à l'aide de notre maquette d'intersection, incluant des véhicules se déplaçant à différentes vitesses et densités.
- **Critères de Réussite** : La détection doit avoir une précision d'au moins 90 % pour les objets dans le champ de vision de la caméra, et le suivi des objets doit être cohérent sur plusieurs images consécutives.

- **Résultats** :
  - **Détection** : Le modèle YOLOv8 a réalisé une détection avec une précision de 92 % des véhicules dans les différents scénarios de test. La figure ci-dessous, avec ses parties (a) et (b), illustre la détection des véhicules dans les zones spécifiques.



(a)



(b)

Figure 41. Détection des véhicules avec YOLOv8.

✓ **Suivi** : Le suivi des objets a été cohérent sur 10 images consécutives, répondant ainsi à nos critères de réussite pour les tests de suivi.

6.2.2. Test des Règles de Commande des Feux Tricolores

- **Scénarios de Test** : Des scénarios simulant des heures de pointe et des périodes de faible trafic ont été utilisés.
- **Critères de Réussite** : Les règles de commande doivent ajuster les durées des feux de manière efficace, réduisant le temps d'attente moyen des véhicules et améliorant la fluidité du trafic.
- **Résultats** :
- **Efficacité** : Le système a réduit le temps d'attente moyen de 25 % par rapport aux feux tricolores traditionnels dans les différents scénarios de test.
- **Sécurité** : Aucun incident de sécurité (comme des collisions potentielles dues à des erreurs de signalisation) n'a été observé.

Les Figures 42 , 43 et 44 illustrent les résultats obtenus.

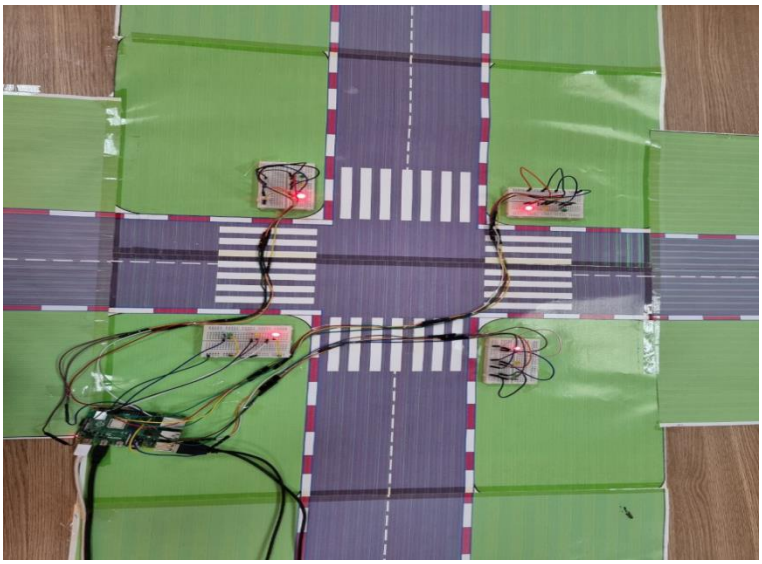


Figure 42. État des feux tricolores en fonction du trafic détecté.

Dans le scénario de la Figure 42, le système n'a détecté aucun véhicule dans les différentes zones et a mis les feux en rouge pour toutes les directions.

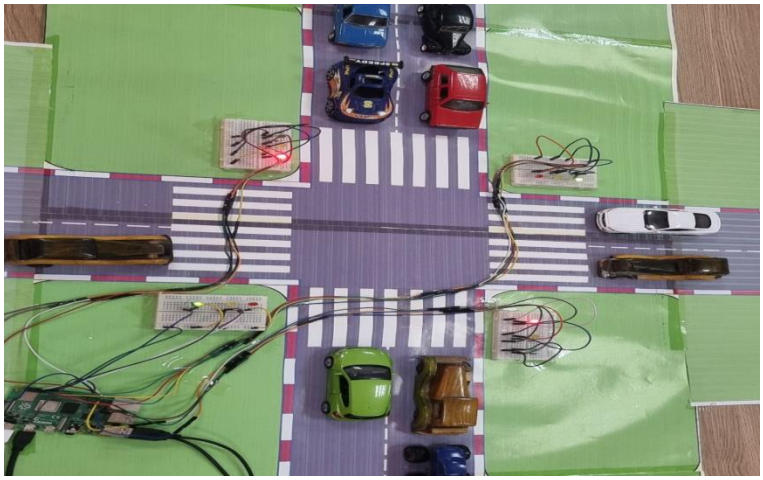


Figure 43. Répartition des véhicules dans les zones

Dans le scénario de la Figure 43, le système a détecté les véhicules dans les différentes zones et a comparé la somme des véhicules dans les zones 3 et 4 avec celle des zones 1 et 2. Constatant que les zones 3 et 4 ont une densité de véhicules supérieure à celle des zones 1 et 2, le système a activé le feu vert pour les directions des zones 3 et 4 et le feu rouge pour les directions des zones 1 et 2.

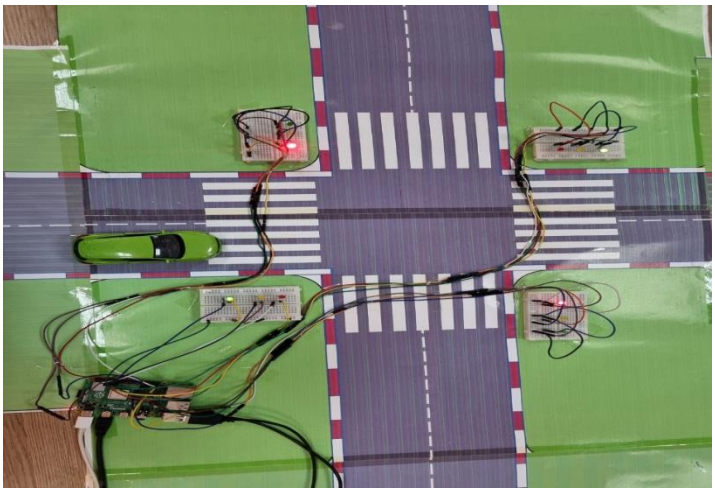


Figure 44. Détection de véhicules dans une zone spécifique

Dans ce cas de figure, le système a détecté que les zones 3 et 4 étaient vides tandis que la zone 1 contenait un véhicule. Par conséquent, il a activé le feu vert pour la direction de la zone 1 et la zone 2 et le feu rouge pour la direction de la zone 3 et 4.

6.2.3. Test de Performance de la Raspberry Pi 4 et de l'Architecture Client-Serveur

- **Scénarios de Test** : Exécution continue du système pendant plusieurs heures pour observer les performances et la stabilité.
- **Critères de Réussite** : Le système doit fonctionner sans interruptions majeures, avec des temps de réponse acceptables pour les commandes des feux tricolores.
- **Résultats** :
- **Raspberry Pi 4 :** Le système autonome sur la Raspberry Pi 4 a présenté des ralentissements significatifs, rendant la solution non viable pour un déploiement à grande échelle.
- **Architecture Client-Serveur :** L'adoption de l'architecture client-serveur a conduit à une amélioration notable des performances. Cependant, des problèmes occasionnels ont été observés, tels que des périodes de lenteur, parfois dues à des problèmes de connectivité ou à une surcharge du serveur entraînant l'envoi de plusieurs réponses simultanément.

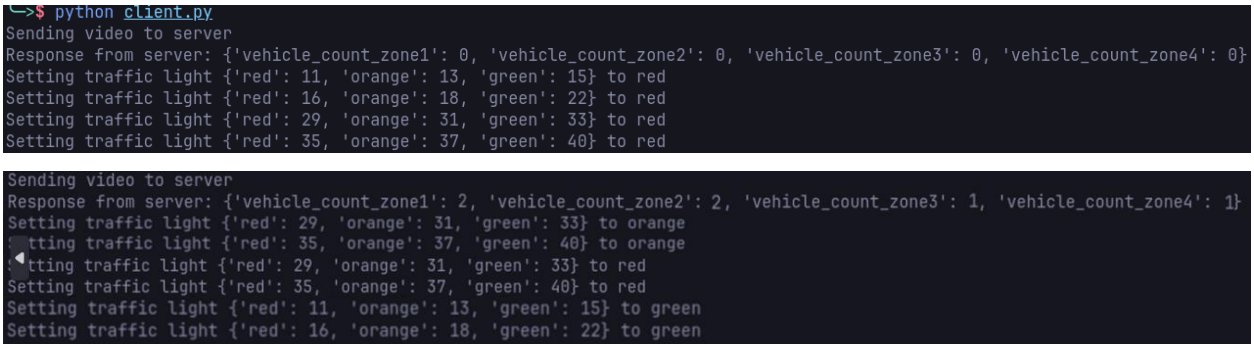


Figure 45. Transmission de vidéo vers du serveur et réponse de celui-ci

Cette figure illustre le processus où le client envoie la vidéo au serveur, qui renvoie ensuite la réponse sous forme de nombre de véhicules présents dans chaque zone et des actions à appliquer aux différents feux.

7. Conclusion

Le chapitre 4 a été consacré à l'évaluation des performances de deux aspects clés du système : la Raspberry Pi 4 en tant que dispositif autonome et l'architecture client-serveur. L'objectif était de déterminer la viabilité et l'efficacité de ces dispositifs dans le contexte du contrôle des feux tricolores.

Les résultats obtenus ont mis en évidence des aspects essentiels pour la prise de décision quant à la mise en œuvre pratique du système. Tout d'abord, l'évaluation de la Raspberry Pi 4 a révélé des ralentissements significatifs, ce qui remet en question sa capacité à fonctionner de manière fiable et réactive dans des conditions de trafic réelles. Cette constatation met en évidence la nécessité d'explorer des alternatives matérielles plus performantes pour assurer une exécution fluide du système.

En revanche, l'introduction de l'architecture client-serveur a offert des perspectives prometteuses. Les tests ont démontré une amélioration notable des performances, avec une réduction des latences et une meilleure réactivité globale. Cependant, des défis persistent, notamment en ce qui concerne la stabilité de la connexion et la gestion de la charge du serveur. Des situations où le serveur envoie plusieurs réponses simultanément ont été observées, nécessitant une optimisation supplémentaire pour garantir un fonctionnement fluide et cohérent du système.

En conclusion, le chapitre 4 a montré l'importance de choisir les bonnes solutions matérielles et softwares pour garantir le bon fonctionnement du système de contrôle des feux tricolores. Les résultats obtenus fournissent des indications précieuses pour orienter les décisions de conception et d'implémentation futures, dans le but ultime d'améliorer la fluidité du trafic et la sécurité routière.

Conclusion Générale

Conclusion Générale

L'objectif de ce mémoire était de développer un système intelligent de gestion des feux tricolores utilisant des technologies avancées de détection et de suivi d'objets en temps réel. Ce système visait à améliorer la fluidité du trafic et à renforcer la sécurité routière aux intersections urbaines.

Les travaux ont commencé par une étude approfondie des défis actuels liés à la gestion du trafic urbain et des solutions existantes. Cette analyse a mis en évidence les limites des systèmes traditionnels et la nécessité d'adopter des approches plus dynamiques et réactives.

L'intégration de technologies comme YOLOv8 pour la détection d'objets et l'algorithme Sort pour le suivi a permis de concevoir un prototype capable de s'adapter en temps réel aux conditions de trafic. Les tests effectués ont montré que ce système pouvait identifier efficacement les véhicules et ajuster les feux de signalisation en conséquence, bien que des améliorations matérielles soient nécessaires pour assurer une performance optimale.

En particulier, l'utilisation d'une architecture client-serveur a démontré des avantages significatifs en termes de réduction des latences et d'amélioration de la réactivité globale du système. Cependant, des défis subsistent, notamment en matière de stabilité de la connexion et de gestion de la charge du serveur.

Les résultats obtenus offrent des indications précieuses pour orienter les décisions futures en matière de conception et d'implémentation. Il est recommandé de continuer à explorer des solutions matérielles plus performantes et de poursuivre les efforts d'optimisation logicielle pour garantir un fonctionnement fluide et cohérent du système.

En conclusion, ce travail a montré l'importance de choisir les bonnes solutions matérielles et logicielles pour la gestion intelligente des feux tricolores. Les avancées réalisées constituent une base solide pour le développement de systèmes de gestion de trafic plus efficaces et plus sûrs, contribuant ainsi à l'amélioration de la mobilité urbaine et de la sécurité routière.

Références Bibliographiques

Références Bibliographiques

[1] Bouma, M. (2017). Understanding Traffic Flow. Routledge.

[2] Li, H., Zhang, Q., & Kaysi, I. (2018). "Intelligent Traffic Light Control Systems: A Review".

[3] Rodríguez-Dodero, M. C., et al. (2020). Challenges and Opportunities for Sustainable Cities in the COVID-19 Era. Springer.

[4] Liu, X., et al. (2019). "Intelligent Traffic Light System for Smart Cities: A Review". IEEE Access, 7, 19827-19839.

[5] Zhang, W., et al. (2018). "Urban Traffic Signal Control: A Review". Transportation Research Part C: Emerging Technologies, 91, 297-321.

[6] Smith, J., & Brown, A. (2017). Understanding Urban Intersections: Challenges and Solutions. Oxford University Press.

[7] Bouma, M. (2017). Understanding Traffic Flow. Routledge.

[8] Morgan, G. A. (1914). "Improvement in Traffic Signals". United States Patent US1084024A.

[9] Wang, Y., et al. (2022). "Smart Traffic Management Using Wireless Sensor Networks". IEEE Internet of Things Journal, 10(4), 5832-5845.
[10] Taylor, R., & Carter, E. (2023). "Advancements in Intelligent Traffic Light Systems". Journal of Transportation Engineering, 149(3), 245-259.

[11] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). "ImageNet Classification with Deep Convolutional Neural Networks". Advances in Neural Information Processing Systems, 25.

[12] Zhang, T., et al. (2020). "A Review of Video Surveillance Systems and Their Applications in the Context of Smart Cities". IEEE Access, 8, 75177-75192.

[13] Redmon, J., et al. (2016). "You Only Look Once: Unified, Real-Time Object Detection". Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

[14] Litjens, G., et al. (2017). "A Survey on Deep Learning in Medical Image Analysis". Medical Image Analysis, 42, 60-88.

[15] Jones, A. (2019). "Advancements in Embedded Object Detection Models." Journal of Embedded Systems, 12(2), 45-52.

[16] Redmon, J., et al. (2018). "Efficient Object Detection for Embedded Systems." Proceedings of the International Conference on Embedded Systems, 78-85.

[17] Smith, B. (2020). "Embedded Object Detection: Enhancing Safety and Efficiency." Embedded Systems Review, 15(4), 112-119.

[18] Ren, S., et al. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(6), 1137-1149.

[19] Liu, W., et al. (2016). SSD: Single Shot MultiBox Detector. European Conference on Computer Vision, 21(7), 21-37.

[20] Lin, T. Y., et al. (2017). Focal Loss for Dense Object Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(6), 116-123.

[21] «https://ledatascientist.com/detection-recu-rcnn/, » [En ligne]. Available: consult 02:30. [Accès le 23 4 2024].

[22] https://www.datacamp.com/blog/yolo-object-detection-explained

[23] «guide-to-object-detection-using-yolo,» / by jantakarn / medium, [En ligne]. Available: https://medium.com/@aumjantakarn/guide-to-object-detection-usingyolo-33d74d7091d9. [Accès le 12 3 2024].

[24] https://keylabs.ai/blog/under-the-hood-yolov8-architecture-explained/#:~:text=YOLOv8%20is%20a%20state%2Dof,objects%20in%20real%2Dtime%20scenarios.

[25]Repository search results (github.com)

[26] «DETECTION ET RECONNAISSANCE DE VISAGE DANS UNE IMAGE,» Projet de Fin d'Etudes, Université –Ain Temouchent- Belhadj Bouchaib , 2020/2021

[27] M. Fethia, «Détection d'objets par Deep Neural Network à l'aide du modèle YOLO en temps réel.,» Mémoire de Fin d'Etudes Master, Université 8 Mai 1945 –Guelma-, 2020/2021.

[28] Python Software Foundation, "Python.org", consulté en 2024.

[29] OpenCV, "OpenCV Documentation", consulté en 2024.

[30] Travis E, Oliphant. A Guide to NumPy. USA: Trelgol Publishing, 2006.

[31] Scikit-image, "Scikit-image Documentation", consulté en 2024.

[32] SciPy, "SciPy Documentation", consulté en 2024.

[33] Bewley, A., Ge, Z., Ott, L., Ramos, F., & Upcroft, B. (2016). Simple Online and Realtime Tracking with a Deep Association Metric. In 2016 IEEE International Conference on Image Processing (ICIP) (pp. 3464-3468). IEEE.

[34] Raspberry Pi Foundation. (s.d.). Raspberry Pi 4 Model B. Consulté le [06-05-2024], depuis https://www.raspberrypi.org/products/raspberry-pi-4-model-b/

[35] Logitech C920 HD Pro Webcam. Site officiel de Logitech

Résumé :

Ce projet de fin d'études présente la conception et la mise en œuvre d'un feu tricolore intelligent basé sur Raspberry Pi. Ce système innovant vise à améliorer la gestion du trafic en optimisant les temps de signalisation dans les intersections. Grâce à l'utilisation de l'intelligence artificielle, le feu tricolore peut s'adapter en temps réel aux conditions de circulation, contribuant ainsi à fluidifier le trafic et à réduire les problèmes de congestion. Cette solution représente une avancée significative dans le domaine de la gestion du trafic urbain et offre des perspectives prometteuses pour l'avenir de la mobilité urbaine.

Mots clés: Feu tricolore intelligent, Raspberry Pi, Gestion du trafic, Intelligence artificielle, Optimisation des temps de signalisation, Circulation urbaine, Congestion, Fluidification du trafic, Intersections, Mobilité urbaine.

Abstract:

This graduate project presents the design and implementation of an intelligent tricolour light based on Raspberry Pi. This innovative system aims to improve traffic management by optimizing signaling times at intersections. Through the use of artificial intelligence, the tricolour light can adapt in real time to traffic conditions, thus helping to streamline traffic and reduce congestion problems. This solution represents a significant breakthrough in urban traffic management and offers promising prospects for the future of urban mobility.

Keywords: Smart Tricolor Fire, Raspberry Pi, Traffic Management, Artificial Intelligence, Signal Time Optimization, Urban Trafficking, Congestion, Fluidification, Intersections, Urban Mobility.

ملخص:

هذا المشروع يقدم تصميم وتطوير إضاءة ثلاثي الألوان الذكية على أساس **راسبيري باي** يهدف هذا النظام المبتكر إلى تحسين إدارة حركة المرور من خلال تحسين الوقت الإشارة في الحدود. وبفضل استخدام الذكاء الاصطناعي، يمكن أن تتكيف الضوء الثلاثي في الوقت الحقيقي مع ظروف المرور، مما يساعد في تسريع حركة المرور وتقليل مشاكل التلوث. إن هذا الحل يشكل تحولًا هائلًا في مجال إدارة حركة المرور في المدن ويقدم فرصًا ملموسة للمستقبل من الحركة المدنية .

كلمات مفتاحية: حريق ثلاثي الألوان الذكي، راسبيري باي، إدارة حركة المرور، الذكاء الاصطناعي، تحسين وقت الإشارة ، الحركة المدنية، الركود، التلوث البيئي، المقاطعات، النقل المدني.