

---

# RAPPORT DU STAGE TECHNIQUE

---

Sujet : la reconnaissance des formes et des angles dans le secteur de l'automobile.



Réalisé par : DOUKKANI Soukaina  
Encadré par : Mr.EL AQQARI Fouad

Année scolaire : 2019/2020

## *Remerciements*

*J'adresse mes remerciements les plus vifs à Madame **BOUILANE Zineb** pour m'avoir donné la chance de poursuivre mon stage technique au sein de la startup Auto360.*

*J'exprime ma très grande gratitude auprès de mon encadrant **Mr. EL AQQARI Fouad** pour son aide gracieux, sa bienveillance et sa disponibilité tout au long de la période du stage.*

*Je remercie mes professeurs et enseignants à l'institut national des postes et télécommunications qui m'ont assuré une formation de haute qualité, un suivi continu et une orientation pertinente vers la réalisation de mes ambitions professionnelles.*

*Un grand merci à ma petite famille qui m'a soutenu financièrement et moralement tout au long de mon parcours académique.*

## *Abstract*

The present report is to show the final outcome of a continuous three months work on object detections and text recognition. This project was done within the company Auto360.

Our final project is part of a real project entitled “detection of shapes and angles for the automotive industry” in which we have employed the deep learning algorithms and computer vision modules to recognize objects within an image and classify them to know either the image contains a car or not and to detect their licence plate and recognize the registration number. For this case, we have used the convolutional neural networks introduced in the YOLO algorithm and the OpenCV methods to classify objects and components into categories.

In this part, we have made the initial operations for the realisation of the Photo 360° that ensures a virtual showroom of cars offering a complete view for the clients and exploring the new technologies enhanced in the realisation of cars.

## *Résumé*

Le présent rapport a pour objectif de montrer le résultat final d'un travail acharné s'étalant sur une période de trois mois dans le contexte de la détection d'objets et la reconnaissance du texte. Ce projet a été réalisé au sein de la startup Auto360.

Dans le projet intitulé « détection de formes et d'angles pour l'industrie automobile » nous avons déployé les algorithmes d'apprentissage profond et les modules de vision par ordinateur pour reconnaître les objets dans une image et les classer afin de détecter les voitures, la plaque d'immatriculation, les roues et déchiffrer le contenu de la plaque. A cet effet, nous avons utilisé les réseaux de neurones convolutifs introduits dans l'algorithme YOLO et les méthodes de la bibliothèque OpenCV pour reconnaître les objets et les composants.

Cette partie concerne la réalisation de la Photo 360° qui assure un showroom virtuel de voitures offrant une vue complète de la voiture pour les clients tout en explorant les nouvelles technologies employées dans sa conception.

## *Table de matières*

Introduction.....	6
Chapitre I : présentation générale du projet.....	7
1. Présentation de l'organisation d'accueil.....	8
2. Etude bibliographique.....	8
2.1. Détection des formes avec OpenCV .....	8
2.2. Les algorithmes de reconnaissance des objets.....	12
2.3. Etude comparative.....	16
Chapitre II : réalisation technique du projet.....	22
1. Processus de détection d'une voiture.....	23
2. Processus de détection de la plaque d'immatriculation.....	25
3. Reconnaissance du contenu de la plaque d'immatriculation..	26
4. Processus de détection des roues.....	27
5. Evaluation de l'algorithme de détection.....	27
Chapitre III : Développement d'une API de détection d'objets.....	35
1. Déploiement de l'API sur Ubuntu VPS.....	36
Conclusion.....	43

## *Introduction*

La détection d'objets est un domaine très actif de la computer vision qui cherche à classer et localiser les régions/zones d'une image ou d'un flux vidéo. Ce domaine est à la croisée de deux autres : la classification d'image et la localisation d'objets. En effet, le principe de la détection d'objets est le suivant : pour une image donnée, on recherche les régions de celle-ci qui pourraient contenir un objet puis pour chacune de ces régions découvertes, on l'extrait et on la classe à l'aide d'un modèle de classification d'image.

De ce fait, il est important de se retourner vers l'apprentissage automatique pour assurer une grande précision pour la détection et la reconnaissance des objets.

Une autre révolution s'ajoutant à la vision par ordinateur est la photo 360° qui peut émettre une vue globale d'un objet. La photo 360° offre une image panoramique qui entoure le point à partir duquel le cliché a été pris.

Dans le secteur automobile, la photo 360° permet de réaliser un showroom virtuel offrant aux visiteurs une vue complète des nouveautés, en plus d'exposer les nouvelles technologies produites dans ce domaine. A cet effet, la photo 360° offre une visite dynamique des produits et attire le maximum de clients potentiels.

Notre travail concerne alors, la détection des formes et la reconnaissance des objets contenus dans une image afin de s'en servir pour déterminer les composantes des voitures et déchiffrer le contenu de la plaque d'immatriculation.

## Chapitre I : présentation générale du projet

Dans la première section de ce chapitre, nous présenterons l'organisme d'accueil, à savoir la startup Auto360, son organisation ainsi que son domaine d'activité. La deuxième section présentera le cadre du projet en se basant sur une étude bibliographique expliquant les différents outils utilisés et les algorithmes employés dans la conception de la solution.

## I. Présentation générale du projet

### 1- Présentation de l'organisme d'accueil

Auto360 est une startup marocaine qui opère dans le secteur automobile. L'organisation assure la réalisation des photo 360° des voitures afin d'organiser un showroom virtuel offrant l'opportunité au futur acheteur de faire le tour de chaque voiture, d'en visiter l'intérieur, d'en découvrir les caractéristiques... sans se déplacer ! Jeux de lumières (feux, clignotants..), effets sonores et visuels (klaxon, moteur, compte-tours...), éléments interactifs (équipements au niveau du tableau de bord) sont autant de liens à la réalité, qui contribuent à la découverte du véhicule et à l'immersion dans son univers. A l'intérieur même de la visite, des appels à l'action permettent enfin de prendre contact, de télécharger une brochure commerciale, voire même de réserver un essai.

### 2- Etude bibliographique

#### 2.1- Détection des formes avec OpenCV

La Computer Vision (CV) est un ensemble de méthodes et de technologies qui permettent d'automatiser une tâche spécifique à partir d'une image. En fait, une machine est capable de détecter, d'analyser et d'interpréter un ou plusieurs éléments d'une image afin de prendre une décision et d'effectuer une action.

OpenCV est une bibliothèque graphique libre largement utilisé dans la computer vision, elle implémente plus de 2500 algorithmes de détection des objets fixes ou en mouvement et permet leurs reconnaissance en se basant sur des modèles de classification tels que SVM . OpenCV fournit un ensemble de plus de 2500 algorithmes de



vision par ordinateur, accessibles au travers d'API. Ce qui permet d'effectuer tout un tas de traitements sur des images (extraction de couleurs, détection de visages, de formes, application de filtres,...). Ces algorithmes se basent principalement sur des calculs mathématiques complexes, concernant surtout les traitements sur les matrices vu que l'image est considérée comme une matrice de pixels.

Dans le processus de détection des formes, nous allons utiliser les fonctionnalités d'OpenCV afin de détecter les composantes d'une voiture.

**1. La lecture des images, des vidéos et des caméras :** la première étape de détection concerne la lecture de l'image, vidéo ou l'activation de la caméra pour visionner les objets en question. Pour se faire, on utilise la fonction prédéfinie `imread()` de OpenCV pour lire une image.

**2. Les fonctions basiques de traitement d'image :** cette étape concerne le filtrage de l'image pour supprimer l'arrière-plan et extraire les objets de l'image. Ceci est fait à l'aide des fonctions suivantes :

- `CvtColor()` : cette fonction permet de modifier les couleurs images et par la suite faciliter le traitement sur une partie précisée de l'image.



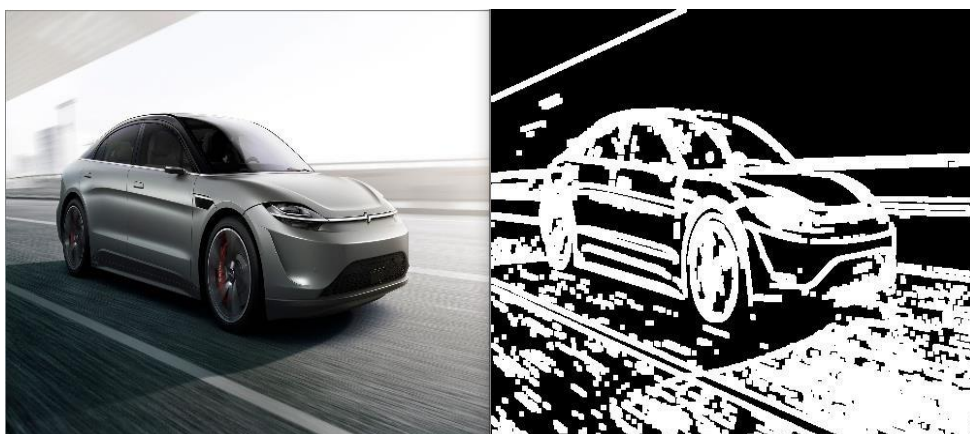
- La fonction `GaussianBlur()` : fournit une image plus foncée que celle émise par la fonction précédente.



- `Canny()` : le filtre de canny permet de supprimer l'arrière plan et retourner une image en binaire qui focalise sur l'objet.



- `imgDilation()` : les contours des objets sont plus visibles et claires.



- `erode()` : cette fonction fournit un contour meilleur de l'objet concerné.



**3. Redimensionnement des images :** la fonction `resize()` permet de modifier les dimensions d'une image ou détecter une partie de l'image initiale.

**4. Identification des formes classiques :** la détection des cercles, rectangles, triangles et d'autres formes géométriques facilitent la détection des objets similaires dans les images, tel que la détection d'un cercle est pareille à détecter une roue dans une voiture. Cette fonctionnalité permet aussi de tracer une droite qui traverse les objets.

**5. Warp perspective :** extraire un objet d'une image en se basant sur les pixels qui présentent les coins d'un objet ayant une forme géométrique.

**6. Collage des images :** une technique utile pour collecter toutes les images en une seule et réduire le temps de traitement de chaque image séparée.

**7. Détection des couleurs dans une image :** modifier les couleurs d'une image pour détecter les objets et optimiser le processus de détection des formes et contours.

**8. Détecter les formes et contours des objets :** utiliser la fonction prédéfinie `findContours()` pour déterminer les contours des objets.

**9. Reconnaissance des objets :** finalement, la dernière phase concerne la reconnaissance des objets contenues dans les images, celle-ci implémente les algorithmes de reconnaissance des objets comme : Haar feature-based cascade classifier, Yolo algorithm et d'autres.

## 2.2- Les algorithmes de reconnaissance des objets

Dans la réalisation de notre projet, nous allons nous référer aux algorithmes de deep Learning afin de traiter les images en input et détecter les objets qu'elles contiennent, ainsi, nous présenterons les différents algorithmes conçus à cet effet.

**2.2.1- Haar feature-based cascade algorithm:** est une méthode efficace de détection d'objets proposée par Paul Viola et Michael Jones dans leur article «Détection rapide d'objets utilisant une cascade boostée de fonctionnalités simples». Il s'agit d'une approche basée sur le machine learning où la fonction en cascade est formée à partir d'un grand nombre d'images positives et négatives. Il est ensuite utilisé pour détecter des objets dans d'autres images.

**2.2.2- Yolo algorithm :** YOLO (You only look once) est venu sur la scène de la computer vision avec le papier séminal 2015 de Joseph Redmon. Connu par sa grande précision, L'algorithme «ne regarde qu'une seule fois» l'image dans le sens où il ne nécessite qu'un seul passage de propagation vers l'avant à travers le réseau neuronal pour faire des prédictions. Après une suppression non maximale (ce qui garantit que l'algorithme de détection d'objet ne détecte chaque objet qu'une seule fois), il génère ensuite des objets reconnus avec les boîtes englobantes.

**2.2.3- Region based convolutional neural networks:** l'algorithme Region based ConvNet applique le principe des réseaux de neurones convolutifs sur des régions de l'image afin de déterminer l'existence d'un objet sur cette partie. Cet algorithme propose par Ross Girshick effectuée, contrairement à l'algo CNN original, le training sur 2000 régions de l'image afin de déterminer les objets présents dans l'image avec précision et leurs affecter des labels déterminant leurs types. L'algorithme fut amélioré plusieurs fois depuis son élaboration ce qui a permis l'apparition des algorithmes Fast R-CNN, Faster R-CNN capables de déterminer les objets dans un temps minimal. Et les algorithmes Mask RCNN dédié pour la segmentation des instances et Mesh R-CNN capable de générer un image 3D à partir d'une image 2D. On se limite dans notre étude aux R-CNN, Fast R-CNN et Faster R-CNN.

R-CNN : À partir d'une image donnée en entrée, le modèle va extraire de l'image les régions les plus susceptibles de contenir un objet. On parle de zones d'intérêts. Pour chacune de ces zones d'intérêts, un ensemble de boîtes englobantes (bounding box) va être généré. Ces boîtes sont classifiées et sélectionnées en fonction de leur probabilité à contenir l'objet. 2000 propositions de régions sont ainsi extraites. Ces régions sont ensuite reçues en entrée par le CNN. Le modèle peut alors détecter dans une image un (ou plusieurs) objet(s), pouvant appartenir à des classes différentes.

Les avantages sont donc de traiter l'image par morceau et non pas toute l'image comme pour un CNN simple et de pouvoir localiser plusieurs objets dans une image.

Fast R-CNN : Avec un modèle R-CNN, chacune des zones d'intérêts proposées est reçue en entrée par le CNN, ainsi l'image de départ subit une convolution par zone proposée. En utilisant un modèle Fast R-CNN,

c'est l'image de départ qui subit cette étape de convolution et c'est sur les features map générées que sont obtenues les propositions de régions susceptibles de contenir l'élément ciblé par le modèle. Cela constitue une amélioration en termes de temps d'exécution et de puissance requise. Faster R-CNN : Le temps de génération des zones d'intérêt est optimisé grâce à l'ajout d'un autre algorithme nommé Region Proposal Networks (RPN) : ce réseau de neurones convolutifs supplémentaire permet de générer directement les propositions de zones. En sortie, le modèle Faster R-CNN produit un label ainsi que les coordonnées de la zone de l'image contenant l'objet détecté (bounding box).

**2.2.4- Kalman filter algorithm:** Le filtre de Kalman estime récursivement l'état de l'objet cible; par conséquent, dans le suivi, c'est une technique utile qui prédit les états des objets en mouvement. Dans l'article original, une solution récursive pour le filtrage optimal linéaire est proposée par R.E. Kalman en 1960. Depuis lors, une recherche approfondie a été effectuée dans divers domaines tels que les systèmes de navigation.

**2.2.5- Single shot multibox algorithm:** Pour mieux comprendre le SSD, commençons par expliquer d'où vient le nom de cette architecture :

- **Single Shot:** cela signifie que les tâches de localisation et de classification des objets se font en une seule passe directe du réseau
- **MultiBox:** c'est le nom d'une technique de régression de boîte englobante développée par Szegedy et al.
- **Détecteur:** le réseau est un détecteur d'objets qui classe également les objets détectés. La technique de régression des boîtes englobantes des SSD est inspirée des travaux de Szegedy sur MultiBox, une méthode pour les propositions de coordonnées de boîtes englobantes rapides

indépendantes de la classe. Fait intéressant, dans le travail effectué sur MultiBox, un réseau convolutionnel de style Inception est utilisé.

**2.2.6- Histogram of oriented gradients:** L'histogramme des gradients orientés (HOG) est un descripteur des caractéristiques. Un descripteur de fonctionnalité est une représentation d'une image - ou de parties d'une image connue sous le nom de correctifs - qui extrait des informations utiles à interpréter par le modèle, telles que des informations cruciales dans l'image telles que des données humaines ou du texte et ignore l'arrière-plan. En tant que tels, les HOG et peuvent être utilisés efficacement dans la détection d'objets.

**2.2.7- Scale invariant feature transform:** Les points clés SIFT des objets sont d'abord extraits d'un ensemble d'images de référence et stockés dans une base de données. Un objet est reconnu dans une nouvelle image en comparant individuellement chaque entité de la nouvelle image à cette base de données et en trouvant des entités concordantes candidates en fonction de la distance euclidienne de leurs vecteurs d'entités. À partir de l'ensemble complet des correspondances, des sous-ensembles de points-clés qui conviennent à l'objet et à son emplacement, son échelle et son orientation dans la nouvelle image sont identifiés pour filtrer les bonnes correspondances. La détermination des grappes cohérentes est effectuée rapidement en utilisant une implémentation de table de hachage efficace de la transformation de Hough généralisée. Chaque groupe de 3 caractéristiques ou plus qui conviennent à un objet et à sa pose est ensuite soumis à une vérification détaillée du modèle et les valeurs aberrantes sont ensuite éliminées. Enfin, la probabilité qu'un ensemble particulier de caractéristiques indique la présence d'un objet est calculée, étant donné la précision de l'ajustement et le nombre de fausses correspondances probables. Les correspondances d'objets qui réussissent

tous ces tests peuvent être identifiées comme correctes avec un niveau de confiance élevé.

Ainsi, il est judicieux de réaliser une étude comparative afin de trouver l'algo le plus adéquat a notre projet.

### 2.3- Etude comparative

-Les contraintes : comme tout algorithme d'apprentissage automatique, la précision et la vitesse de prédiction de l'algorithme sont les deux contraintes principales qui doivent être optimises pour juger qu'un algorithme est puissant et précis.

-Les critères de comparaison : afin de choisir l'algorithme le plus puissant pour la détection des objets, nous allons comparer les algorithmes en se basant leurs modes de fonctionnement incluant leurs complexités. De plus, le taux de précision de l'algorithme et son temps d'exécution.

Algorithme	Complexité	Précision	Temps d'exécution
Haar cascade classifieurs	L'algorithme effectue une détection de l'image intégrale émise dans l'entrée. Il se fait sur trois phases	Précision moyenne, l'algorithme est sensible aux rotations des objets et changement des angles de vue de l'objet.	31 ms



YOLO algorithm		Yolo peut détecter les objets dans une image en la regardant seulement une fois, ce qui optimise la vitesse de détection des objets et explique son utilisation dans le domaine de vision par ordinateur. l'algo le plus recommande pour la détection des objets en temps réel.	L'algorithme de yolo applique sur le dataset COCO a atteint une précision de 65.7%	YOLO est considéré l'algorithme le plus rapide en terme de détection des objets.
CNN	RCNN	<p>-Extraire 2000 régions pour chaque image sur la base d'une recherche sélective.</p> <p>-La détection des objets se fait sur trois modules :</p> <p>1-L'application de CNN pour l'extraction des entités.</p> <p>2-Classificateur SVM linéaire pour identifier des objets</p> <p>3-Modèle de régression pour resserrer les boîtes englobantes.</p>	L'algorithme a atteint le taux de précision de 68% lorsqu'il est appliqué sur Pascal VOC dataset. Et seulement 38% de précision lorsqu'il est appliqué sur le dataset COCO.	40-50 secondes
	Fast RCNN	Il utilise également la recherche sélective comme méthode de proposition pour trouver les régions	68.2% de précision pour le dataset Pascal VOC	2 secondes

		d'intérêt, ce qui est un processus lent et long.		
	Faster RCNN	<p>Le réseau ne regarde pas l'image complète d'un seul coup, mais se concentre sur des parties de l'image de manière séquentielle.</p> <p>Cela crée deux complications:</p> <ul style="list-style-type: none"> <li>-L'algorithme nécessite de nombreux passages à travers une seule image pour extraire tous les objets</li> <li>-Étant donné que différents systèmes fonctionnent l'un après l'autre, les performances des systèmes plus avancés dépendent de la performance des systèmes précédents.</li> </ul>	78.5% en précision applique sur le même dataset	0.2 secondes
Kalman filter algorithm		L'algorithme est performant pour la reconnaissance des objets en mouvement.	72%	200 ms
SSD		L'algorithme prédit mal les petits objets. Dans SSD, les petits objets ne peuvent être détectés que dans les	70%	53.5 ms

	<p>couches de résolution supérieure (couches les plus à gauche).</p> <p>Le SSD présente une erreur de localisation inférieure par rapport au R-CNN mais davantage d'erreur de classification concernant des catégories similaires.</p>		
HOG	<p>Le processus de détermination des objets très long vu que l'algorithme divise l'image en très petites cellules et calcule pour chaque cellule l'histogramme des directions du gradient ou des orientations des contours pour les pixels à l'intérieur de cette cellule.</p>	73.5%	300 ms
SIFT	<p>L'idée général de cette méthode est de transformer une image en vecteurs de caractéristiques, lesquels doivent être dans l'idéal invariants aux transformations géométriques (rotation,</p>	65%	400 ms

	<p>mise à l'échelle), et dans une moindre mesure invariants à l'illumination.</p> <p>Il s'agit de détecter des points remarquables (ou clés), qui vont permettre d'identifier un objet. La détection de ces points donne lieu à la mise en place des vecteurs de caractéristiques dont les composantes sont propres au point considéré.</p>		
--	---	--	--

En se basant sur l'étude comparative, on remarque que les algorithmes sont similaires en terme de précision et de temps d'exécution. Le seul critère qui peut faire la différence entre les algos est leurs modes de fonctionnement et complexités.

On estime que les algorithmes YOLO et Haar cascade sont plus faciles à déployer et compréhensibles. Ainsi que ces algorithmes ne sont primordiaux dans notre projet vu qu'ils seront utilisé dans la partie initiale pour filtrer les inputs et ne garder que les images contenant une voiture.

Ainsi, pour bien mener notre mission, il est recommandé d'utiliser l'algorithme YOLO vu son temps d'exécution rapide et la facilité de son implémentation.

## Conclusion :

Dans ce premier chapitre, nous avons présenté le contexte général du projet dans lequel nous avons cité les différentes fonctionnalités de la bibliothèque OpenCV ainsi que lister les algorithmes les plus pertinents dans la détection d'objets pour enfin conclure par une étude comparative menant à proposer l'algorithme qui sera utilisé par la suite dans l'élaboration de la solution.

## Chapitre II : réalisation technique du projet

Le deuxième chapitre est dédié à la conception technique de la solution, ceci inclut dans une première phase les processus de détection de la voiture, la plaque d'immatriculation et les roues, ainsi que la reconnaissance du contenu de la plaque d'immatriculation. Dans la deuxième phase, nous allons évaluer l'algorithme de détection et calculer sa précision.

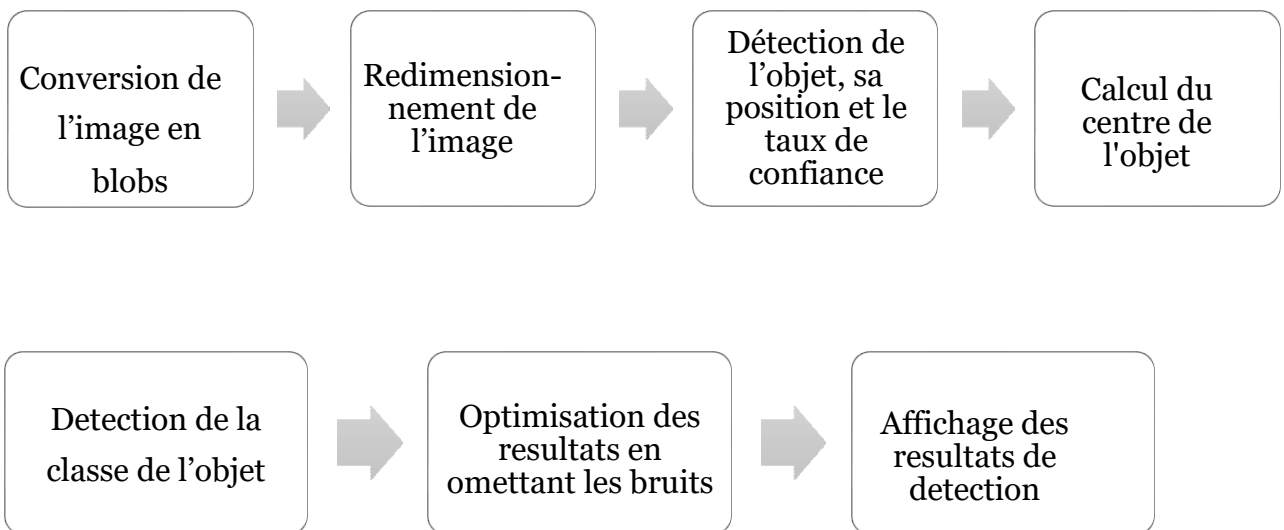
## 1. Processus de détection de la voiture

Dans notre code, on va utiliser les bibliothèques numpy et Opencv.

On commence par importer le code de l'algorithme YOLO V3 a l'aide de la fonction readnet()

Le modèle est appliqué sur le dataset COCO, donc on va importer le fichier coco.names contenant les classes des objets que l'algorithme est capable de détecter.

Le processus de détection de l'algorithme est le suivant :



1- **Conversion de l'image en blobs** : le réseau de neurones ne peut pas détecter les features dans l'image directement, tout d'abord on doit la convertir en blobs pour qu'il puisse extraire les objets contenus dans l'image.

Pour se faire, on va utiliser la fonction blobfromimage()

2- **Redimensionnement de l'image** : YOLO accepte trois dimensions d'image :

- 320\*320 : l'image est trop petite ce qui affecte la précision
- 609\*609 : l'image est plus grande, la précision est optimale mais la vitesse d'exécution est réduite.

- 416\*416 : image moyenne

3- **Détection de l'objet** : l'algorithme appliqué sur les blobs arrive à détecter l'objets en se basant sur sa position et le taux de confiance élevé,

4- **Centre d'objet** : une fois l'objet est détecté, on va calculer son centre (center\_x, center\_y) et préciser ses coordonnées (x,y, width, height) afin de tracer les bounding boxes autour de l'objet détecté.

5- **La classe de l'objet** : maintenant , on affecte a l'objet sa catégorie en se basant sur 'classe\_id'

6- **Omission des bruits** : il se peut qu'on trouve plusieurs boxes pour le même objet, c'est pourquoi on utilise la fonction NMSBoxes() pour enlever ce bruit et garder un seul bounding box.

7- **Affichage des résultats de détection** : on va utiliser la fonction labels() pour afficher le bounding box et la catégorie de l'objet.

Input :





Output :



## 2-Processus de détection de la plaque d'immatriculation

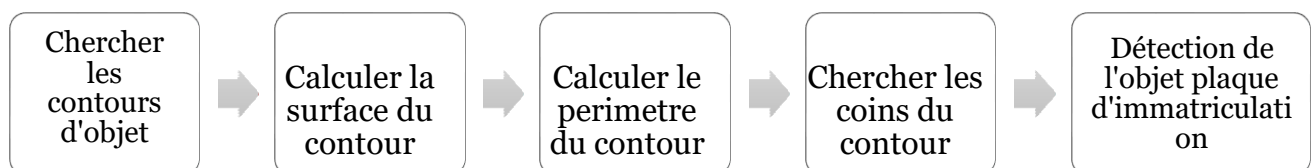
Afin de détecter la plaque d'immatriculation , on va se servir des fonctionnalités de la bibliothèque OpenCV.

### *I. Préparation de l'image pour le processing*

Cette phase préliminaire est importante pour préparer l'image au traitement, on va tout d'abord supprimer les couleurs de l'image a l'aide de la fonction `cvtColor()`, puis on utilise la fonction `GaussianBlur()` pour rendre l'image moins claire afin de réduire le bruit. Finalement on utilise la fonction `Canny()` pour détecter les contours des objets.

### *II. Détection des objets*

Dans cette phase, on va utiliser la fonction `getContours()`. Le processus de détection d'objet est le suivant :



- 1-Chercher les contours des objets : on utilise la fonction `findContours()`.
- 2-La surface du contour : on calcule 'area' afin de filtrer les contours trouvés et garder les contours qui peuvent référer à des objets.
- 3-Calculer le périmètre : on utilise la fonction `arcLength()`
- 4-Chercher les coins du contour : la fonction `approxPolyDP()` permet de trouver les coins du contour et par la suite détecter la forme.
- 5-Détection de l'objet : on sait que la plaque est sous forme d'un rectangle et que  $height \ll width$ , cad si le nombre de coins est égale à 4 alors l'objet est une plaque d'immatriculation.

### 3- Reconnaissance du contenu de la plaque d'immatriculation

Après avoir détecté la composante plaque d'immatriculation dans la voiture, nous allons lire son contenu et déterminer la préfecture d'émission de la plaque.

Pour se faire, on va utiliser la bibliothèque Pytesseract qui permet de lire du texte à partir d'une image.

Ainsi, on doit installer Tesseract-OCR (Optical Character Recognition), un logiciel de reconnaissance du texte dans une image. Le rôle de Tesseract-OCR est de convertir une image bi-dimensionnelle contenant des caractères et des mots, en texte lisible et facile à être utiliser.

Ensuite, nous avons créé une liste des identifiants des plaque avec les préfectures correspondantes.

De plus, la fonction `image_to_string()` de pytesseract permet de détecter le texte contenu dans l'image.

Output :

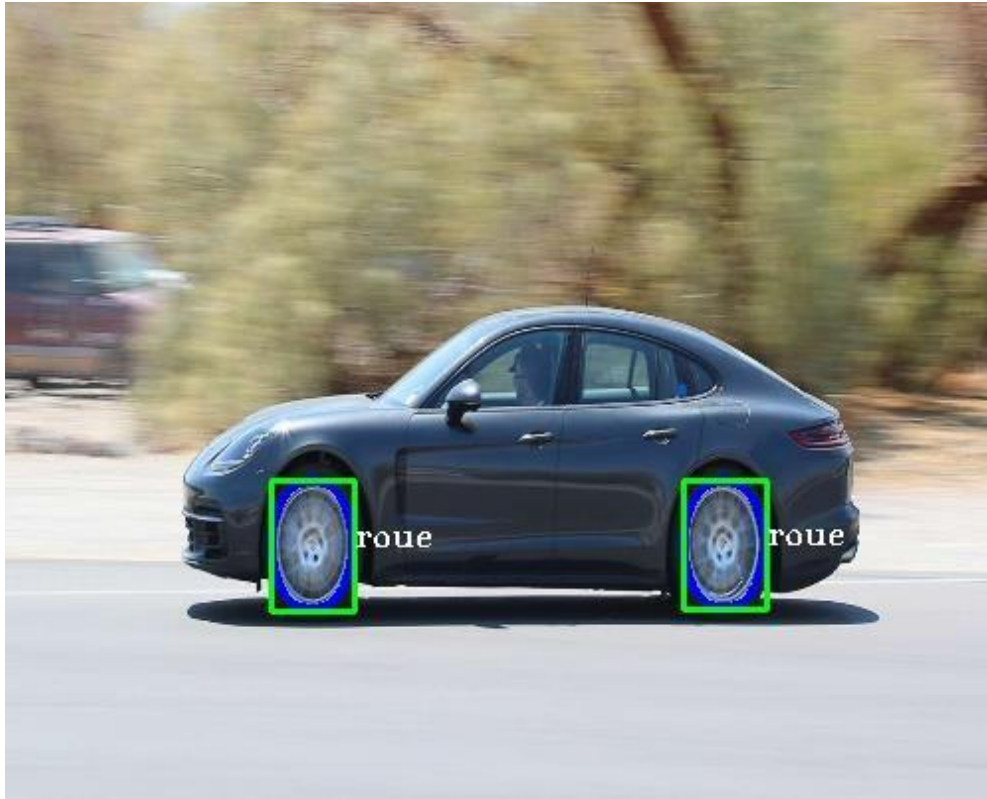


```
Détection-plaque d'immatriculation x
C:\Users\soukaina\PycharmProjects\YOLO-object-detection-algorithm\venv\Scripts\python
detection du contenu de la plaque 68858|A|72
le contenu de la plaque ['6', '8', '8', '5', '8', '|', 'A', '|', '7', '2']
l'identifiant de la préfecture : 72
La préfecture d'émission de la plaque est Casablanca - Aïn chok
|
```

### 4-Processus de détection des roues

Le processus de détection des roues est pareil à celui de détection de la plaque d'immatriculation. La seule différence est dans le nombre de coins vu que la forme 'cercle' possède un nombre de coins très grand .

Output :



## 5-Evaluation de l'algorithme

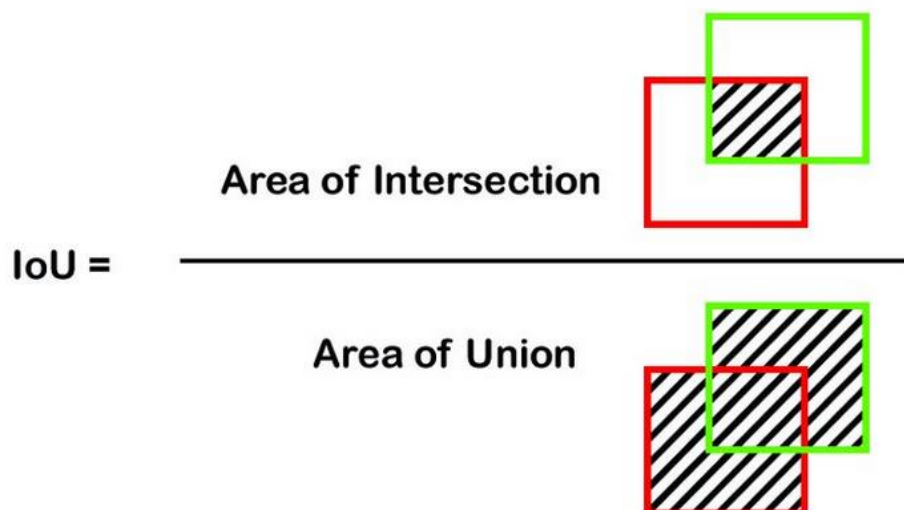
Le principe général d'évaluation des algorithmes de détection d'objets est de comparer le bounding Box réalisé manuellement avec celui établi par l'algorithme afin de calculer une métrique d'évaluation nommée Intersection sur Union (IsU).

La précision de cette métrique permet de déterminer les TP,FP,TN,FN et par la suite déterminer la valeur de précision et rappel (sensibilité) pour obtenir la valeur de ***mean average precision (mAP)*** .

On commence par définir chaque métrique :

- Intersection sur Union : désigne la différence entre deux bounding boxes, l'un établi manuellement et l'autre prédit par l'algorithme.





Le rectangle rouge désigne le bounding box établi manuellement à l'aide de LabelImg, tandis que le rectangle en vert désigne le bounding box prédit par l'algorithme YOLO.

L'image suivante précise les bounding box entourant l'objet 'Voiture'.



Pour une valeur donnée de Threshold :

- Si  $IsU > Threshold$  : l'algorithme génère une bonne prédiction de l'objet, donc la détection sera classifiée comme un **True positive (TP)**
- Si  $IsU < Threshold$  : l'algorithme ne réussit pas à détecter la voiture dans l'image, donc la détection sera classifiée comme un **False positive (FP)**
- Si  $IsU = 0$ , l'image contient l'objet 'Voiture', cependant l'algorithme n'a rien prédit donc la détection sera classifiée comme un **False negative (FN)**
- Si  $IsU = 0$ , l'image ne comporte pas l'objet 'voiture' et l'algorithme n'a rien prédit, donc la détection sera classifiée comme un **True negative (TN)**

Ainsi, nous avons déterminé toutes les valeurs nécessaires pour évaluer l'algorithme.

Après la précision des FP, TP, FN, TN, il est temps de calculer la précision et le rappel.

-Précision : précise à quel point la prédiction est exacte en calculant le pourcentage des prédictions correctes (TP) sur la somme des prédictions (TP+FP)

$$Precision = TP / (TP + FP)$$

-Rappel : désigne la capacité du modèle à trouver tous les points d'intérêt. En d'autres termes, c'est la mesure de la qualité de notre modèle pour découvrir tous les aspects positifs.

$$Rappel = TP / (TP + FN)$$

Il nous reste de calculer la précision interpolée qui s'agit simplement de la valeur de précision la plus élevée pour certaines valeurs de rappel. Par exemple, si nous avons la même valeur de rappel 0,2 pour trois valeurs de précision différentes de 0,87, 0,76 et 0,68, la valeur de la précision

interpolée pour les trois valeurs de rappel sera la plus élevée parmi ces trois valeurs, soit 0,87.

$$precision\_interpolee = \max(\sum Precision(rappel))$$

Finalement, nous allons calculer la valeur de mean Average precision (mAP). Pour se faire, on va calcule la somme des précisions interpolées dans 11 niveaux différents du rappel entre 0 et 1.

$$mAP = \frac{1}{11} * (AP_{recall(0)} + AP_{recall(0.2)} + AP_{recall(0.4)} + AP_{recall(0.8)})$$

### Obtention des dimensions des Bounding Box :

Comme discuté précédemment, l'évaluation de l'algorithme nécessite la précision des coordonnées des bounding box.

A cet effet, nous avons lister les valeurs (x,y,width,height) des bounding box établis à l'aide de l'algorithme YOLO dans liste\_prediction.

Ainsi, pour obtenir les valeurs saisis manuellement, on va se servir de l'outil LabelImg largement utilise pour délimiter les objets dans les images et les nommer.

L'outil LabelImg permet de préciser l'objet dans l'image et le nommer dans le format YOLO.

Il retourne un fichier .txt contenant les valeurs suivantes :

```
object-id center_x center_y width height
```

Avec Object\_id : concerne l'ID de l'objet entoure dans l'image

Center\_x , Center\_y : détermine les coordonnées du centre de l'objet. Les valeurs sont normalisées entre 0 et 1 en les divisant par les valeurs de width et height de l'image respectivement.

Width, height : représentent la largeur et longueur du Bounding box, les valeurs sont aussi normalisées entre 0,1 par division par la largeur et longueur de l'image.

La figure suivante présente le processus de création du bounding box autour de l'objet 'voiture'



LabelImg enregistre les coordonnées du Bounding box dans un fichier txt comme suit :

```
15 0.481667 0.550336 0.863333 0.711409
```

Donc, pour déterminer les valeurs originales des (center\_x, center\_y, width, height), nous allons multiplier les valeurs reçues par width et height de l'image.

Ainsi, la fonction lire\_valeurs\_reelles(fichier) permet de lire le fichier comportant les coordonnées des bounding box de toutes les images depuis le fichier 'file'.

De ce fait, nous disposons des valeurs des bounding box nous permettons de calculer IsU et par la suite préciser mAP.



On va utiliser la fonction `Calcul_ISU_Metrique (Val_reelle, Val_predite)` pour obtenir les valeurs de la zone d'intersection et la zone d'union afin de calculer leurs différences.

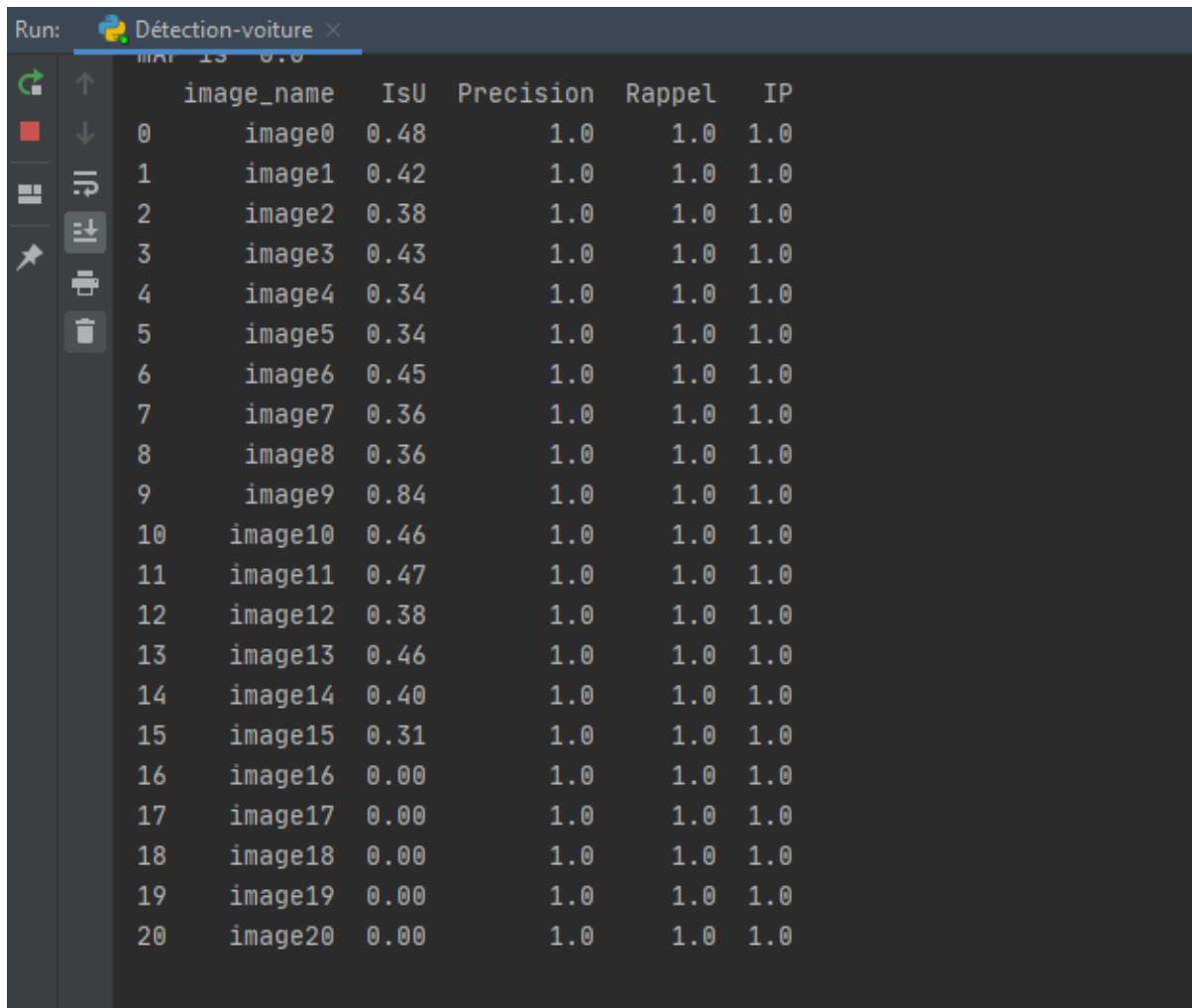
On va créer une table d'évaluation contenant les colonnes (image, précision, rappel, précision interpolée) .

On applique le code sur un dataset de 21 image dont 16 contiennent une voiture et 5 ne la contiennent pas, la matrice de confusion résultante est la suivante :

```
TP 16  
FP 0  
TN 5  
FN 0
```

Effectivement, l'algorithme a pu détecter avec succès les images contenant une voiture et ceux qui la contiennent pas.

Ainsi, la table d'évaluation englobant tous les éléments précités est comme suit :



	image_name	IsU	Precision	Rappel	IP
0	image0	0.48	1.0	1.0	1.0
1	image1	0.42	1.0	1.0	1.0
2	image2	0.38	1.0	1.0	1.0
3	image3	0.43	1.0	1.0	1.0
4	image4	0.34	1.0	1.0	1.0
5	image5	0.34	1.0	1.0	1.0
6	image6	0.45	1.0	1.0	1.0
7	image7	0.36	1.0	1.0	1.0
8	image8	0.36	1.0	1.0	1.0
9	image9	0.84	1.0	1.0	1.0
10	image10	0.46	1.0	1.0	1.0
11	image11	0.47	1.0	1.0	1.0
12	image12	0.38	1.0	1.0	1.0
13	image13	0.46	1.0	1.0	1.0
14	image14	0.40	1.0	1.0	1.0
15	image15	0.31	1.0	1.0	1.0
16	image16	0.00	1.0	1.0	1.0
17	image17	0.00	1.0	1.0	1.0
18	image18	0.00	1.0	1.0	1.0
19	image19	0.00	1.0	1.0	1.0
20	image20	0.00	1.0	1.0	1.0

Donc la précision de l'algorithme est optimale.

Ainsi, nous avons établi le code python nécessaire pour détecter la voiture et ses composantes. Le chapitre suivant sera consacré au développement d'une API de détection d'objets basée sur le code réalisé précédemment et son déploiement dans le serveur virtuel Ubuntu VPS 18.04 LTS.

## Chapitre III : Développement d'une API de détection d'objets

Dans ce dernier chapitre, nous allons développer une API de détection d'objets et par la suite la déployer sur le serveur Ubuntu.

## 1- Déploiement de l'API de détection.

Dans cette phase, nous allons utiliser le framework Flask pour créer l'application web en python.

### Etape1 : installation des prérequis

A l'instar des projets de machine learning, le déploiement du code sur un serveur dédié est indispensable pour le rendre accessible depuis tous les dispositifs informatiques. Ainsi, nous allons utiliser le serveur web Apache2 et le module mod\_wsgi qui constitue une interface facilitant l'hébergement des applications développées avec python sur le serveur web.

Tout d'abord, on commence par mettre à jour le serveur Ubuntu

*sudo apt-get update*

*sudo apt-get upgrade*

Ainsi, on va installer Apache 2.4.29 et vérifier son état.

*sudo apt install apache2*

On vérifie l'état du serveur à l'aide de la ligne de commande suivante :

*sudo service apache2 status*

```
ubuntu@vps-0689a54b:~$ sudo service apache2 status
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
   Drop-In: /lib/systemd/system/apache2.service.d
            └─apache2-systemd.conf
   Active: active (running) since Mon 2020-09-28 22:26:39 UTC; 21min ago
     Process: 12694 ExecStop=/usr/sbin/apachectl stop (code=exited, status=0/SUCCESS)
     Process: 31881 ExecReload=/usr/sbin/apachectl graceful (code=exited, status=0/SUCCESS)
     Process: 12699 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Main PID: 12714 (apache2)
      Tasks: 55 (limit: 2260)
   CGroup: /system.slice/apache2.service
           └─12714 /usr/sbin/apache2 -k start
             └─12717 /usr/sbin/apache2 -k start
               └─12718 /usr/sbin/apache2 -k start

Sep 28 22:26:39 vps-0689a54b systemd[1]: Starting The Apache HTTP Server...
Sep 28 22:26:39 vps-0689a54b systemd[1]: Started The Apache HTTP Server.
ubuntu@vps-0689a54b:~$
```

On installe mod\_wsgi en utilisant :

*sudo apt-get install libapache2-mod-wsgi-py3*

Et on l'active en utilisant : *sudo a2enmod wsgi*

On vérifie l'état de mod\_Wsgi en accédant au dossier  
/usr/lib/apache2/modules via la commande :

*ldd mod\_wsgi.so*

```
ubuntu@vps-0689a54b:~$ cd /usr/lib/apache2/modules/
ubuntu@vps-0689a54b:/usr/lib/apache2/modules$ ldd mod_wsgi.so
linux-vdso.so.1 (0x00007fff577d4000)
libpython3.6m.so.1.0 => /usr/lib/x86_64-linux-gnu/libpython3.6m.so.1.0 (0x00007f1f8d465000)
libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007f1f8d246000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f1f8ce55000)
libexpat.so.1 => /lib/x86_64-linux-gnu/libexpat.so.1 (0x00007f1f8cc23000)
libz.so.1 => /lib/x86_64-linux-gnu/libz.so.1 (0x00007f1f8ca06000)
libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007f1f8c802000)
libutil.so.1 => /lib/x86_64-linux-gnu/libutil.so.1 (0x00007f1f8c5ff000)
libm.so.6 => /lib/x86_64-linux-gnu/libm.so.6 (0x00007f1f8c261000)
/lib64/ld-linux-x86-64.so.2 (0x00007f1f8dd4a000)
ubuntu@vps-0689a54b:/usr/lib/apache2/modules$
```

Comme mentionné dans la figure, mod\_wsgi utilise la version 3.6.9 de python qui est la version appliquée par défaut sur Ubuntu 18.04, ainsi, nous allons installer les bibliothèques nécessaires pour cette version spécifique.

A cet effet, on commence par installer pip3 qui va nous permettre d'installer les packages pour python3.6 :

*sudo apt install python3-pip*

on installe aussi l'environnement de développement pour python3 via

*sudo apt install python3-dev*

maintenant on va installer Flask, numpy, opencv et pytesseract

Flask 0.12.2: *pip3 install flask*

Numpy 1.13.3: *pip3 install numpy*

OpenCV 4.4.0.42: *sudo apt install libopencv-dev*

*Pip3 install opencv-python*

Pytesseract 4.0.0: *pip3 install pytesseract*

Installation de Tesseract-ocr pour la langue arabe

*sudo apt install tesseract-ocr-ara*

Etape 2 : Création de l'API sur le serveur

Dans une deuxième étape, nous allons transférer le projet vers le serveur en utilisant FTP.

Tout d'abord, nous allons placer l'API dans le répertoire /var/www à l'aide de la commande suivante :

```
cd /var/www  
sudo mkdir DetectionAPP  
cd DetectionAPP  
sudo mkdir APP_code  
cd APP_code  
sudo mkdir static templates YOLO  
cd static  
sudo mkdir uploads detections plaques
```

Avant de transférer le projet vers Ubuntu, on va modifier les permissions pour donner l'accès à FTP pour modifier dans le serveur :

```
sudo chmod -R a+rwx /var/www/DetectionAPP/APP_Code
```

Ainsi, la structure du répertoire est la suivante :

```
tree /var/www/DetectionAPP/APP_Code
```

```
ubuntu@vps-0689a54b:/var/www/DetectionAPP/APP_Code$ tree
.
├── API.py
├── __pycache__
│   └── Training_model.cpython-36.pyc
├── static
│   ├── detections
│   ├── plaques
│   └── uploads
├── templates
│   ├── about.html
│   ├── index.html
│   └── output.html
├── Training_code.py
└── YOLO
    ├── coco.names
    ├── yolov3.cfg
    └── yolov3.weights

7 directories, 9 files
```

On va tester l'installation du code en exécutant l'API sur le serveur local :

*sudo python3 API.py*

```
ubuntu@vps-0689a54b:/var/www/DetectionAPP/APP_Code$ sudo python3 API.py
* Running on http://127.0.0.1:4000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 382-573-204
```

### Etape3 : configuration de Apache2 et mod\_wsgi

Nous allons créer un nouvel hôte virtuel pour héberger l'API sur le serveur web.

Tout d'abord, on va accéder au répertoire `/etc/apache2/sites-available` pour créer le fichier de configuration:

*sudo nano /etc/apache2/sites-available/APP.conf*

Dans le fichier APP.conf nous allons définir le virtual host

```
<VirtualHost *:80>
    ServerName 51.210.45.30
    ServerAdmin admin
    WSGIScriptAlias / /var/www/DetectionAPP/APP.wsgi
    <Directory /var/www/DetectionAPP/APP_Code/>
        Order allow,deny
        Allow from all
    </Directory>
    Alias /static /var/www/DetectionAPP/APP_Code/static
    <Directory /var/www/DetectionAPP/APP_Code/static/>
        Order allow,deny
        Allow from all
    </Directory>
    ErrorLog ${APACHE_LOG_DIR}/error.log
    LogLevel warn
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

On va activer l'hôte virtuel en utilisant la commande suivante :

*Sudo a2ensite APP.conf*

On passe maintenant à créer le fichier APP.wsgi dans le répertoire /var/www/DetectionAPP pour exécuter l'API sur Apache2

*sudo nano /var/www/DetectionAPP/APP.wsgi*

```
#!/usr/bin/python3
import sys
import logging
logging.basicConfig(stream=sys.stderr)
sys.path.insert(0, "/var/www/DetectionAPP/APP_Code/")
sys.path.insert(0, "/var/www/DetectionAPP/APP_Code/static")
from API import app as application
```

Finalement, on va redémarrer le serveur pour appliquer les modifications .

*sudo service apache2 restart*



## Résultat final



### Détection des objets

La détection des objets dans les images est une problématique qui a affronté l'Humain depuis une longue période.

Aujourd'hui grâce aux algorithmes de machine learning et à la vision par ordinateur, la reconnaissance des objets est désormais possible et réalisable.

Ainsi, dans le contexte de la photo360, nous avons développé une application web capable de reconnaître les voitures, leurs plaques d'immatriculation ainsi que déchiffrer le matricule à l'aide de l'OCR.

D'où l'application de l'algorithme YOLO et les modules de la bibliothèque OpenCV pour détecter, reconnaître et localiser les voitures dans les images.

Continuer



### API DE DETECTION D'OBJETS

Veuillez insérer une image

Browse... No file selected.

Envoyer



### Processus de détection des objets

Image d'origine





Image après détection



### Detection de la plaque d'immatriculation



le contenu de la plaque d'immatriculation est : 68858 FT 172)

## *Conclusion*

Répondre aux problématiques liés à la détection , reconnaissance et segmentation des objets n'est pas aussi facile que l'on pense, surtout lorsque les objets détectés sont très petits et que l'on doit s'en servir pour répondre à une autre problématique bien distincte liée à la détection des angles.