

Traitement de video

2022

Lane Detection for Autonomous vehicle

Réalisé par :

ANASS EL HALLANI
OTHMANE ELAZRI
YOUNES EL BELGHITI

Encadrée par :

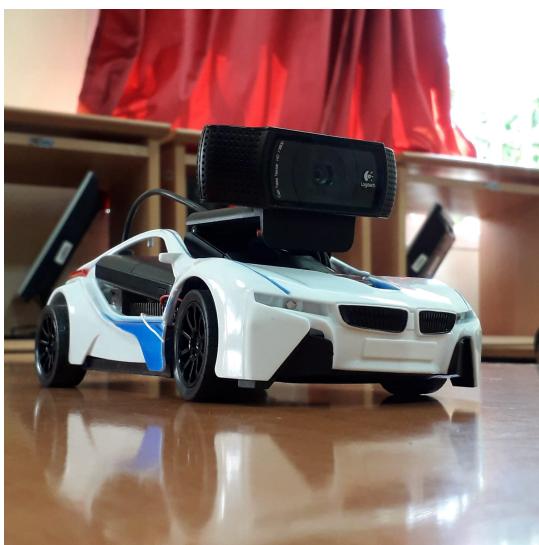
Pr.FATIMA ZAHRA EL BIACH
Pr.ZINEB El ABIDI

Résumé

Dans le futur proche, les véhicules autonomes devraient :

- améliorer la fluidité du trafic
- réduire la consommation
- alléger la navigation
- réduire de carburant et la criminalité

Malgré les divers avantages potentiels de l'utilisation de l'automatisation des véhicules, il y a des problèmes non résolus de sécurité et responsabilité. Par conséquent, Notre projet a proposé une détection de voie basée sur Raspberry Pi et la langage du programmation Python pour améliorer la sécurité et la responsabilité des voitures autonomes. La détection de voie est réalisée en capturant d'abord l'image à partir d'une caméra vidéo et la convertie en échelle de gris. Ensuite, un processus de filtrage du bruit pour l'image grise est effectué à l'aide d'un filtre gaussien, puis nous appliquons quelques techniques de détection des contours en utilisant des méthodes de base, comme dérivée première ou seconde pour voir le changement brutal d'un pixel à l'autre. L'effet de la modification automatique du seuil sur l'image améliore notre algorithme.



keyword : Détection de voie, Détection du contour, Véhicule autonome, Raspberry Pi, Python

Sommaire

0.1	Introduction	5
1	Etat de l'art	6
1.1	Espaces de couleur	6
1.1.1	RGB vers niveau du gris :	6
1.2	Réduction du bruit :	7
1.3	Détection du contour :	8
1.4	Masque d'une image :	8
1.5	conclusion	9
2	La partie hardware et Software :	10
2.1	Hardware utilise pour le creation de circuit	10
2.1.1	Raspberry Pi	10
2.1.2	Servomoteur	11
2.2	Software utilisé en programmation	12
2.2.1	OpenCV	12
2.2.2	NumPy	13
2.2.3	Python	13
2.2.4	Microsoft Visual Studio	14
2.3	Conclusion :	14
3	Experimentation et discussions des résultats	15
3.1	Étapes d'Algorithme	15
3.1.1	Preparation des frames	15
3.1.2	Reduction de bruit	16
3.1.3	Trover moyenne de frame	16
3.1.4	Masque de frame	17
3.1.5	Seuillage binaire	18
3.1.6	Detection de contour	18
3.1.7	Autocorrection	19
3.1.8	Traitement de frame	20
3.1.9	Main	20
3.2	les résultats obtenus	21

3.3 conclusion	21
Table des annexes	22
Annexe A Creation de model de detection d'objet	23
A.1 Introduction	23
A.2 Convolutional neural networks	23
A.2.1 Definition	23
A.2.2 Sofware utiliser : YOLOv5	24
A.3 Étapes pour construire le modèle	25
A.3.1 Étapes 1 : L'ensemble de données du projet	25
A.3.2 Étapes 2 : Convertir les annotations au format YOLOv5	26
A.3.3 Étapes 3 : train le modèle YOLOv5	26
3.4 Resultat de training	27

Table des figures

1	détection du lignes	5
1.1	RVB vers niveau du gris	7
1.2	Debruitage par filtre gaussien blur	7
1.3	Detection du contour	8
1.4	Masquer une image	9
2.1	Image du Raspberry Pi 2	10
2.2	Raspberry Pi 3 Modèle B + GPIO Bloc 40 broches et brochage de l'en-tête PoE	11
2.3	servomoteur	12
2.4	logo opencv	12
2.5	logo NumPy	13
2.6	Image de paython	13
2.7	Visual Studio Code	14
3.1	Preparation de frame	15
3.2	Frame Original	16
3.3	Frame du RGB au Gray	16
3.4	fonction de reduction de bruit	16
3.5	Frame de depart	16
3.6	Frame sans bruit	16
3.7	Calcul de moyenne	17
3.8	Fonction de masque	17
3.9	Tracage de masque	17
3.10	Frame sans le masque	18
3.11	Frame avec le masque	18
3.12	Seuillage	18
3.13	Seuillage sans le masque	18
3.14	Seuillage avec le masque	18
3.15	Detection de contour	19
3.16	Detection de contour sans le masque	19
3.17	Detection de contour avec le masque	19

3.18 Fonction pour detecter les variations d'angle	19
3.19 Ligne detecter	20
3.20 Servo moteur pour rotation	20
3.21 Traitement global de frame	20
3.22 Main	21
3.23 Resultat du traitement	21
A.1 Comparison between image classification, object detection and instance segmentation.	23
A.2 Image du processus de détection d'objet (citation de YOLOv5)	24
A.3 Darknet-53	25
A.4 Classes des panneaux de signalisation	25
A.5 exemple d'annotation au format YOLO où l'image contient deux objets différents.	26
3.6 Precision	27
3.7 Recall	27
3.8 Matrice de confision	27
3.9 Resultat	27
3.10 Resultat	27

0.1 Introduction

Ces derniers temps, le nombre de véhicules routiers a énormément augmenté grâce aux technologies avancées de l'industrie automobile et très précisément à la disponibilité de tarifs réduits.

Comme le temps qu'un homme utilise une voiture est considérablement plus long, le conducteur se sent fatigué dans le système de conduite passive actuel qu'un conducteur donne et prend une commande. Avec cette croissance remarquable, le nombre d'accidents augmentent.

Le développement de voiture autonome permet d'assister le conducteur de différentes manières afin de garantir sa sécurité, ce qui préserve également la sécurité des autres conducteurs et des piétons.

La détection des lignes de voie est un composant essentiel pour les voitures autonomes et également pour la vision par ordinateur en général. Ce concept est utilisé pour décrire la trajectoire des voitures autonomes et pour éviter le risque de se retrouver dans une autre voie.

L'objectif principal de Notre système est de détecter les lignes de voie en temps réel utilisant Raspberry Pi pendant le processus de conduite. Grâce à ces fonctionnalités, le système peut guider et alerter les conducteurs pour prévenir un danger.

Notre project est composée de trois principaux chapitres. Tout d'abord,nous commençons par une introduction pour montrer la vue générale de notre projet, puis nous allons exposer quelques rappels et définitions et informations sur la détection de voie , ensemble de matériels et logiciel qui permet de mettre notre projet en marche . Ensuite, nous allons nous intéresser à la réalisation de Notre projet avec raspberry Pi 3 utilisant la langage python.



FIGURE 1 – détection du lignes

Chapitre 1

Etat de l'art

Ce chapitre décrit les techniques de détection de ligne sur la voie qui contribuent à la sécurité routière, car la sécurité est une priorité absolue pour tous les systèmes de détection de voie. La plupart des accidents de voiture sont causés par l'incapacité du conducteur à contrôler la trajectoire du véhicule. Par conséquent, nous avons décidé de mettre en œuvre ce projet dans le but de sauver la vie des conducteurs et de minimiser le taux d'accidents.

1.1 Espaces de couleur

- Les couleurs sont très nombreuses (6 couleurs est juste une convention)
- Besoin de représenter les couleurs dans un espace générateur.
- Toute couleur peut être représentée comme une combinaison linéaire de trois primaires c_1, c_2, c_3
- Beaucoup d'espaces de couleur : RGB, HSV, YUV, Lab,...

1.1.1 RGB vers niveau du gris :

Une image en niveaux de gris est simplement une image dans laquelle les seules couleurs sont des nuances de gris. La raison pour différencier ces images de tout autre type d'image couleur est que moins d'informations doivent être fournies pour chaque pixel. En fait, une couleur "grise" est une couleur dans laquelle les composantes rouge, verte et bleue ont toutes la même intensité dans l'espace RVB, et il suffit donc de spécifier une seule valeur d'intensité pour chaque pixel, par opposition aux trois intensités nécessaires pour spécifier chaque pixel dans une image en couleur.[8]

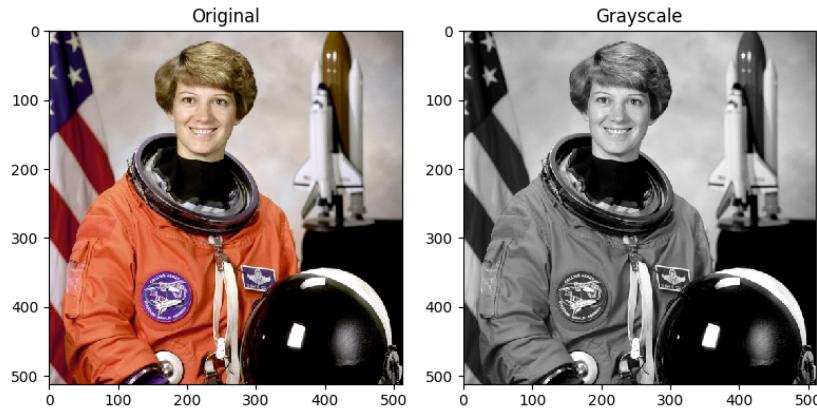


FIGURE 1.1 – RVB vers niveau du gris

1.2 Réduction du bruit :

Le bruit est un problème mondial réel pour tous les systèmes et la vision par ordinateur ne fait pas exception. Les algorithmes développé doit être tolérant au bruit ou le bruit doit être éliminé.

La présence de bruit dans notre système entravera la détection correcte des bords, de sorte que la suppression du bruit est une condition préalable à une détection efficace des bords .

Le débruitage est le processus de suppression ou de réduction du bruit ou des artefacts de l'image. Le débruitage rend l'image plus claire et nous permet de voir clairement les détails les plus fins de l'image. Cela ne modifie pas directement la luminosité ou le contraste de l'image, mais en raison de la suppression des artefacts, l'image finale peut sembler plus lumineuse. Dans ce processus de débruitage, nous choisissons une boîte 2D et la glissons sur l'image. L'intensité de chaque pixel de l'image originale est recalculée à l'aide de la boîte.[2]



FIGURE 1.2 – Debruitage par filtre gaussien blur

1.3 Détection du contour :

Les limites des voies sont définies par un contraste net entre la surface de la route et les lignes peintes ou certains type de surface non pavée. Ces contrastes saisissants sont les bords de l'image. Par conséquent, les détecteurs de bord sont très important pour déterminer l'emplacement les frontières de la voie.

En traitement d'image et en vision par ordinateur, on appelle détection de contours les procédés permettant de repérer les points d'une image matricielle qui correspondent à un changement brutal de l'intensité lumineuse. Ces changements de propriétés de l'image numérique indiquent en général des éléments importants de structure dans l'objet représenté. Ces éléments incluent des discontinuités dans la profondeur, dans l'orientation d'une surface, dans les propriétés d'un matériau et dans l'éclairage d'une scène.

La détection des contours dans une image réduit de manière significative la quantité de données en conservant des informations qu'on peut juger plus pertinentes. Il existe un grand nombre de méthodes de détection des contours de l'image mais la plupart d'entre elles peuvent être regroupées en deux catégories. La première recherche les extremaums de la dérivée première, en général les maximums locaux de l'intensité du gradient. La seconde recherche les annulations de la dérivée seconde, en général les annulations du laplacien ou d'une expression différentielle non linéaire. Exemples d'algorithmes de détection de contour : Canny, Sobel, Laplacian, etc. [3]

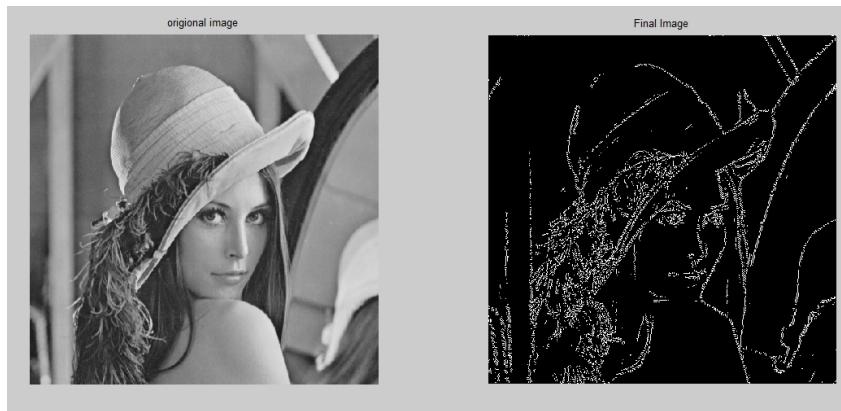


FIGURE 1.3 – Detection du contour

1.4 Masque d'une image :

Le masquage est utilisé dans le traitement d'image pour produire la région d'intérêt, ou simplement la partie de l'image qui nous intéresse. Nous avons tendance à utiliser des opérations au niveau du bit pour le masquage car cela nous permet de supprimer les parties de l'image dont nous n'avons pas besoin.[4]

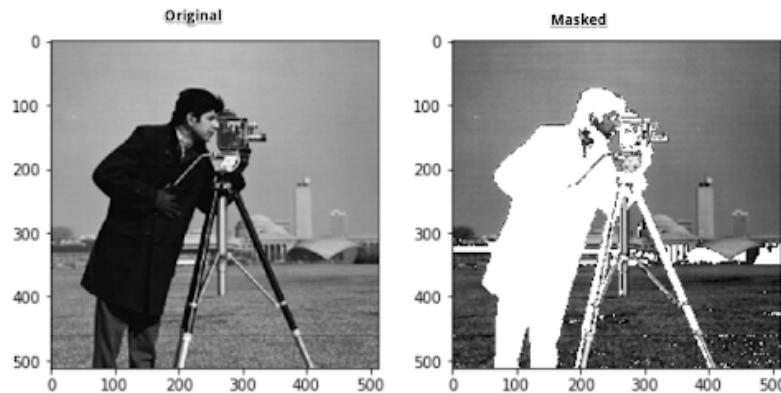


FIGURE 1.4 – Masquer une image

1.5 conclusion

Dans ce chapitre nous présentons une étude générale sur les différents etapes et techniques pour la detection du lignes.

Chapitre 2

La partie hardware et Software :

La réalisation du projet nécessite un ensemble de matériels qui permet de mettre en marche plusieurs actions qui sont orientés à partir de notre projet. Plus nombreux frameworks sont disponibles, et chacun présente ses propres avantages. matplotlib,OpenCv,Numpy. . . .

2.1 Hardware utilise pour le creation de circuit

2.1.1 Raspberry Pi

Definition de Raspberry :

Le Raspberry Pi n'est rien d'autre qu'un ordinateur réduit à sa plus simple expression : une unique carte à processeur ARM un poil plus grande qu'une carte de crédit. Son intérêt : rendre l'informatique abordable et accessible à tous (comptez entre 5 et 35 € suivant les modèles) tout en encourageant l'apprentissage de la programmation informatique et de ses différents langages. Même si vous n'êtes pas un grand passionné de hardware et de programmation informatique, il y a fort à parier que vous avez déjà entendu parler du Raspberry Pi. Et en effet, comme nous allons le voir, ce nano-ordinateur novateur n'est pas seulement destiné à un public averti. [6]



FIGURE 2.1 – Image du Raspberry Pi 2

Caractéristique :

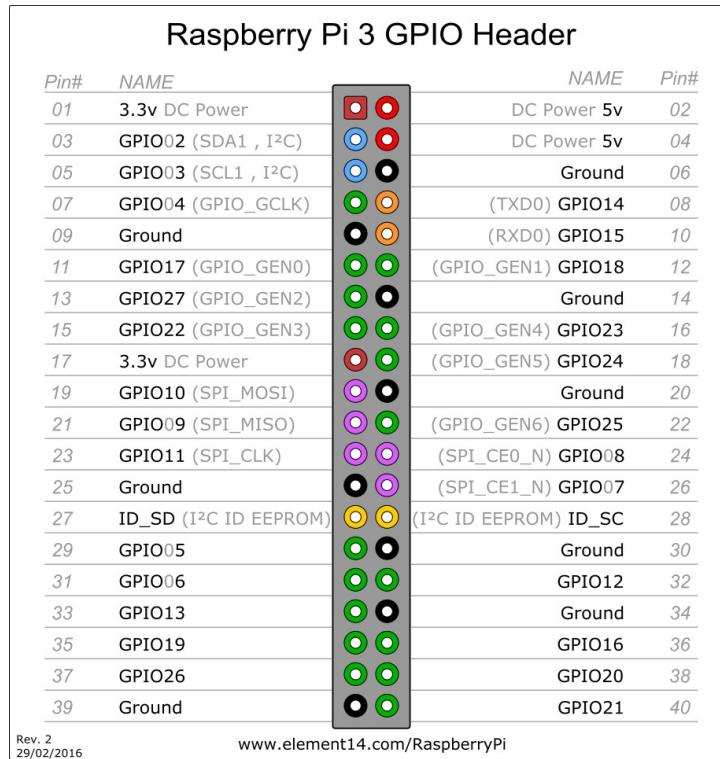


FIGURE 2.2 – Raspberry Pi 3 Modèle B + GPIO Bloc 40 broches et brochage de l'en-tête PoE

2.1.2 Servomoteur

Présentation

Un servomoteur est un système qui a pour but de produire un mouvement précis en réponse à une commande externe. C'est un actionneur (système produisant une action) qui intègre l'électronique, la mécanique et l'automatique.

Un servomoteur est capable d'attendre des positions prédéterminées dans les instructions qui lui ont été donné, puis de les maintenir. Un servomoteur est composé de :

- * Un dispositif électronique d'asservissement.
- * Un axe dépassant hors du boîtier avec différents bras ou roues de fixation
- * Un potentiomètre (faisant fonction de diviseur résistif) qui génère une tension variable proportionnelle à l'angle de l'axe de sortie.
- * Un réducteur en sortie de ce moteur diminuant la vitesse et augmentant le couple. Techniquement, ce langage servira surtout pour le scripting et l'automatisation (interaction avec les navigateurs web).
- * Un micro moteur à courant continu.

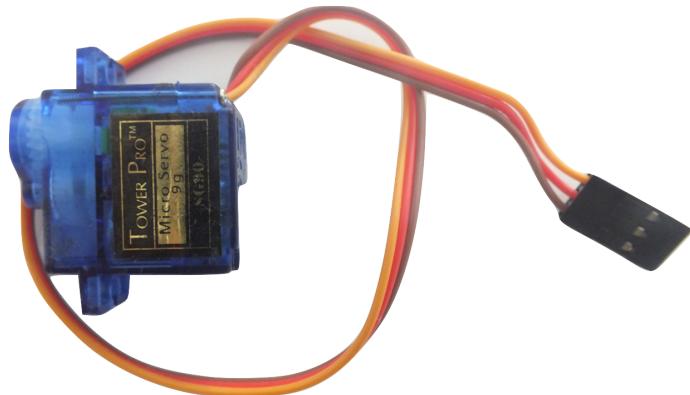


FIGURE 2.3 – servomoteur

Dans l'univers Arduino, le servomoteur est essentiellement utilisé dans les applications robotiques. Le servomoteur possède trois fils de connexion pour pouvoir fonctionner, deux fils servent à son alimentation, le dernier étant celui qui reçoit le signal de commande :

- * Rouge : pour l'alimentation positive (4.5V à 6V en général).
- * Noir ou marron : pour la masse (0V).
- * Orange, jaune, blanc, ... : entrée du signal de commande.

2.2 Software utilisé en programmation

2.2.1 OpenCV

OpenCV signifie "Open Source Computer Vision" est une bibliothèque de logiciels de vision par ordinateur et d'apprentissage automatique inventée par Intel en 1999. OpenCV possède des interfaces C++, Python, Java et MATLAB et prend en charge Windows, Linux, Android et Mac OS. OpenCV a été écrit nativement en C++ et possède une interface basée sur des modèles qui fonctionne de manière transparente avec les conteneurs STL. [1]



FIGURE 2.4 – logo opencv

2.2.2 NumPy

NumPy est une bibliothèque pour langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux. Plus précisément, cette bibliothèque logicielle libre et open source fournit de multiples fonctions permettant notamment de créer directement un tableau depuis un fichier ou au contraire de sauvegarder un tableau dans un fichier, et manipuler des vecteurs, matrices et polynômes. [5]



FIGURE 2.5 – logo NumPy

2.2.3 Python

Python est le langage de programmation open source le plus employé par les informaticiens. Ce langage s'est propulsé en tête de la gestion d'infrastructure, d'analyse de données ou dans le domaine du développement de logiciels. En effet, parmi ses qualités, Python permet notamment aux développeurs de se concentrer sur ce qu'ils font plutôt que sur la manière dont ils le font. Il a libéré les développeurs des contraintes de formes qui occupaient leur temps avec les langages plus anciens. Ainsi, développer du code avec Python est plus rapide qu'avec d'autres langages. [9]

Les principaux utilisations de Python par les développeurs sont :

- * la programmation d'applications
- * la création de services web
- * la génération de code
- * la métaprogrammation. Techniquement, ce langage servira surtout pour le scripting et l'automatisation (interaction avec les navigateurs web).



FIGURE 2.6 – Image de python

2.2.4 Microsoft Visual Studio

Microsoft Visual Studio est une suite de logiciels de développement pour Windows conçu par Microsoft. La dernière version s'appelle Visual Studio 2010. Visual Studio est un ensemble complet d'outils de développement permettant de générer des applications Web ASP.NET, des Services Web XML, des applications bureautiques et des applications mobiles. Visual Basic, Visual C++, Visual C et Visual J utilisent tous le même environnement de développement intégré (IDE, Integrated Development Environment), qui leur permet de partager des outils et facilite la création de solutions faisant appel à plusieurs langages. Par ailleurs, ces langages permettent de mieux tirer parti des fonctionnalités du Framework .NET, qui fournit un accès à des technologies clés simplifiant le développement d'applications Web ASP et de Services Web XML grâce à Visual Web Developer.

[7]

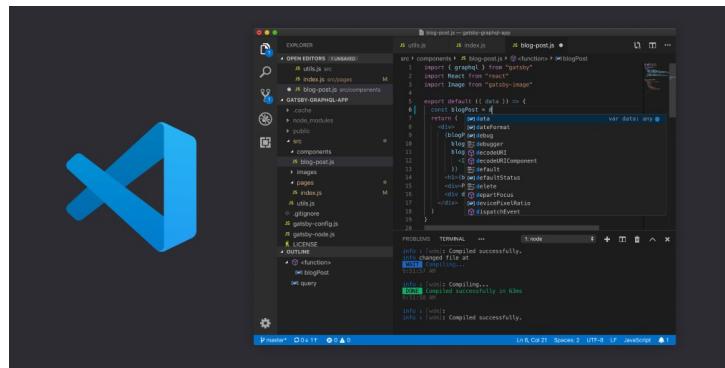


FIGURE 2.7 – Visual Studio Code

2.3 Conclusion :

Ce chapitre donne un aperçu et des détails pour l'étude conceptuelle avant de réaliser le prototype de notre projet en deux parties :

- Partie d'appareils qu'une présentation d'appareils électroniques du système globale.
- Partie du logiciel qui représente les principaux plateformes, langages, frameworks et environnement de programmation

Chapitre 3

Experimentation et discussions des résultats

3.1 Étapes d'Algorithme

Dans ce partie , nous allons construire un algorithme capable de détecter les lignes de voie en temps réel utilisant Raspberry Pi pendant le processus de conduite.

pour construire notre algorithme, nous avons suivi ces étapes ci-dessous

3.1.1 Preparation des frames

Premièrement, on prépare chaque frame pour faciliter le traitement, donc c'est mieux du convertir image du RGB au Gray pour avoir moins de données dans le traitement, et on change le type de liste du python liste a numpy.array pour avoir moins d'espace utilisé au temps de traitement.



```
def prepareImg(fram):
    img = cv.cvtColor(fram, cv.COLOR_BGR2GRAY)
    img = np.array(img)
    return img
```

FIGURE 3.1 – Preparation de frame



FIGURE 3.2 – Frame Original



FIGURE 3.3 – Frame du RGB au Gray

3.1.2 Reduction de bruit

Ce qu'on appelle du bruit sur une photo n'est évidemment pas audible. En photographie, le bruit numérique est un phénomène causé par un éclairage insuffisant lors de la prise de vue. Et on l'élimine par un filtre de lissage pour réduire le bruit de sel poivre.

```
def reduirBruit(img):
    return cv.bilateralFilter(img, 5, 50, 50)
```

A screenshot of a dark-themed code editor window. It shows a single Python function definition. The function is named 'reduirBruit' and takes one parameter 'img'. It returns the result of applying the 'cv.bilateralFilter' function to 'img' with parameters 5, 50, and 50. The code is written in Python, using standard conventions like color-coded keywords and syntax highlighting.

FIGURE 3.4 – fonction de reduction de bruit



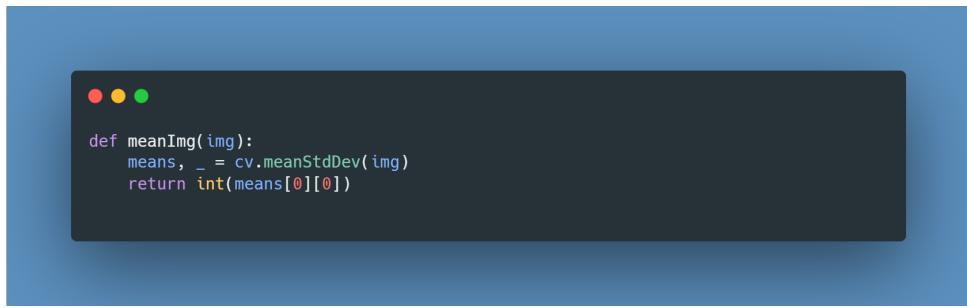
FIGURE 3.5 – Frame de depart



FIGURE 3.6 – Frame sans bruit

3.1.3 Trouver moyenne de frame

Pour rendre notre programme compatible à toutes les variances de lumière durant la journée on a pris en compte cet variance, donc on a récupéré la moyenne de chaque frame pour avoir une grande flexibilité.



```
def meanImg(img):
    means, _ = cv.meanStdDev(img)
    return int(means[0][0])
```

FIGURE 3.7 – Calcul de moyenne

3.1.4 Masque de frame

Pour avoir une grande précision durant le traitement de chaque frame, on a masqué les parties de frame qu'on n'a pas besoin, et aussi pour minimiser le temp de traitement le maximum possible, donc on gagne à la fois le temps et la précision.



```
def mask(img, v):
    mask = np.zeros_like(img)
    chaine_count = 1
    match_mask_color = (255,) * chaine_count
    cv.fillPoly(mask, v, match_mask_color)
    mask_img = cv.bitwise_and(img, mask)
    return mask_img
```

FIGURE 3.8 – Fonction de masque

On choisit le masque compatible avec notre cas de traitement via un vecteur qui trace ou il faut être le masque sur le frame.



```
def maskImg(img):
    v = [(-50, img.shape[0]), (img.shape[1]/2-60, img.shape[0]/1.6-30),
        (img.shape[1]/2+20, img.shape[0]/1.6-30), (img.shape[1]+50, img.shape[0])]
    return mask(img, np.array([v], np.int32),)
```

FIGURE 3.9 – Tracage de masque



FIGURE 3.10 – Frame sans le masque



FIGURE 3.11 – Frame avec le masque

3.1.5 Seuillage binaire

Le seuillage binaire pour la détection rapide et facile des contours de frame.

```
def seuilImg(img, mean, masked_img):
    img = cv.split(img)[0]
    (., newImg) = cv.threshold(masked_img, mean + 55, 255, cv.THRESH_BINARY)
    return newImg
```

FIGURE 3.12 – Seuillage



FIGURE 3.13 – Seuillage sans le masque



FIGURE 3.14 – Seuillage avec le masque

3.1.6 Détection de contour

la détection de contours est le processus permettant de repérer les points d'une image matricielle qui correspondent à un changement brutal de l'intensité lumineuse. Ces changements de propriétés de l'image numérique indiquent en général des éléments importants de structure dans l'objet représenté. Ces éléments incluent des discontinuités dans la profondeur, l'orientation d'une surface, les propriétés d'un matériau et dans l'éclairage d'une scène.

```

def contour(fram, newImg):
    contours, hierarchy = cv.findContours(image=newImg, mode=cv.RETR_TREE,
method=cv.CHAIN_APPROX_NONE)
    image_draw = fram.copy()
    cv.drawContours(image=image_draw, contours=contours, contourIdx=-1, color=(0,
0, 0), thickness=5, lineType=cv.LINE_AA)
    return image_draw

```

FIGURE 3.15 – Détection de contour



FIGURE 3.16 – Détection de contour sans le masque



FIGURE 3.17 – Détection de contour avec le masque

3.1.7 Autocorrection

Cette fonction détecté la distance entre le pixel central et les pixels de l'extrémité des lignes intérieur, pour obtenir la direction et la quantité de rotation du servo pour ajuster la direction de voiture la maintenir dans la même voie.

```

def autoCorrection(axe=460, ligne=400):
    m = (np.where(newImg[ligne,:]==255))
    pose = [ (m[0][i],m[0][i-1]) for i in range(len(m[0])) if m[0][i]-m[0][i-
1]>300]
    disDroit = np.abs(pose[0][0]-axe)
    disGauche = np.abs(pose[0][1]-axe)
    if disGauche > disDroit:
        direction, valeurRoutation = 0, np.abs(disDroit-disGauche)
    elif disDroit > disGauche:
        direction, valeurRoutation = 1, np.abs(disDroit-disGauche)
    else:
        direction, valeurRoutation = -1, np.abs(disDroit-disGauche)
    return direction, valeurRoutation

```

FIGURE 3.18 – Fonction pour detecter les variations d'angle

On cherche les pixels où existent les valeurs max pour calculer la différence entre l'axe central et l'extrême.



FIGURE 3.19 – Ligne detecter

Cette fonction fait tourner le servo moteur par des valeurs bien pressée pour corriger la direction de la voiture sur la voie.

```
def servo(pin, sleep, angle):
    servoMotor = AngularServo(pin, min_pulse_width=0.0006, max_pulse_width=0.0023)
    servoMotor.angle = angle
    time.sleep(sleep)
```

FIGURE 3.20 – Servo moteur pour rotation

3.1.8 Traitement de frame

Le processus passe par toutes les fonctions pour effectuer la détection de ligne avec une grande précision et un temps de traitement aussi court afin que nous puissions effectuer la détection de ligne en temps réel

```
def traitement(frame):
    ligne = 400
    img = prepareImg(frame)
    img = reduirBruit(img)
    mean = meanImg(img)
    mask = maskImg(img)
    seuillage = seuilImg(img, mean, mask)
    direction, valeurRoutation = autoCorrection(seuillage, ligne=ligne)
    angle = np.arccos(valeurRoutation/(img.shape[0]-ligne))*180/np.pi
    servo(18, 0.01, angle if direction else -angle)
    contourImg = contour(frame, seuillage)
    return contourImg
```

FIGURE 3.21 – Traitement global de frame

3.1.9 Main

Dans la main on démarre la boucle de traitement pour qu'on peut détecter les linge de la route.



```
def main():
    cap = cv.VideoCapture(0)
    while True:
        ret, frame = cap.read()
        if ret == True:
            cv.imshow('frame', traitement(frame))
            if cv.waitKey(1) & 0xFF == ord('q'):
                break
        else:
            break

    cap.release()
    cv.destroyAllWindows()
```

FIGURE 3.22 – Main

3.2 les résultats obtenus

enfin, après toutes les étapes suivies, nous arrivons au point, nous détectons des lignes comme celles-ci apparaissent sur la photo.



FIGURE 3.23 – Resultat du traitement

3.3 conclusion

Dans ce projet de détection de ligne de voie, nous utilisons OpenCV. Avant de détecter les lignes de voie, nous avons masqué les objets restants, puis identifié la ligne. Le projet a réussi à détecter clairement les lignes de voie dans les flux vidéo.

Ce projet nous a aidé à comprendre certains outils de traitement vidéo et comment les utiliser dans notre vie quotidienne.

Annexes

Annexe A

Creation de model de detection d'objet

A.1 Introduction

Dans ce partie , nous allons construire un modèle de réseau neuronal profond baser sur yolov5 capable de détecter et classer les panneaux de signalisation pendant le processus de conduite en différentes catégories. Avec ce modèle, nous sommes capables de lire et de comprendre les panneaux de signalisation qui sont une tâche très importante pour tous les véhicules autonomes.

A.2 Convolutional neural networks

A.2.1 Definition

Les réseaux de neurones convolutifs (CNN) sont de l'intelligence artificielle algorithmes basés sur des réseaux de neurones multicouches qui apprennent caractéristiques des images, étant capable d'effectuer plusieurs tâches comme classification, détection et segmentation d'objets.

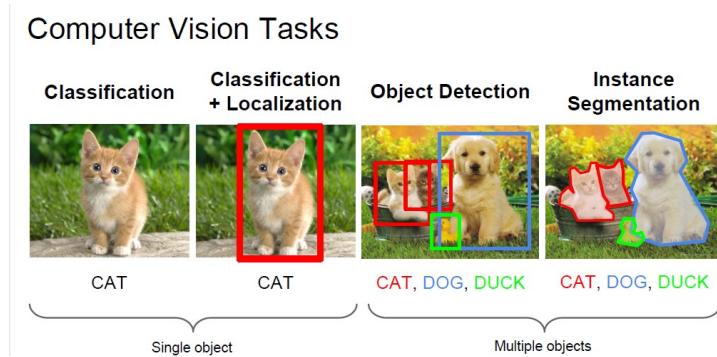


FIGURE A.1 – Comparison between image classification, object detection and instance segmentation.

A.2.2 Sofware utiliser : YOLOv5

YOLO (You only look once) est un système de détection d'objets en temps réel à la pointe de la technologie. Il s'agit d'un algorithme de reconnaissance et de localisation d'objets basé sur un réseau neuronal profond. Sa plus grande caractéristique est qu'il fonctionne très vite. Par exemple, si vous entrez une image, le système affichera les objets qu'il contient et la position de chaque objet (le cadre rectangulaire contenant l'objet).

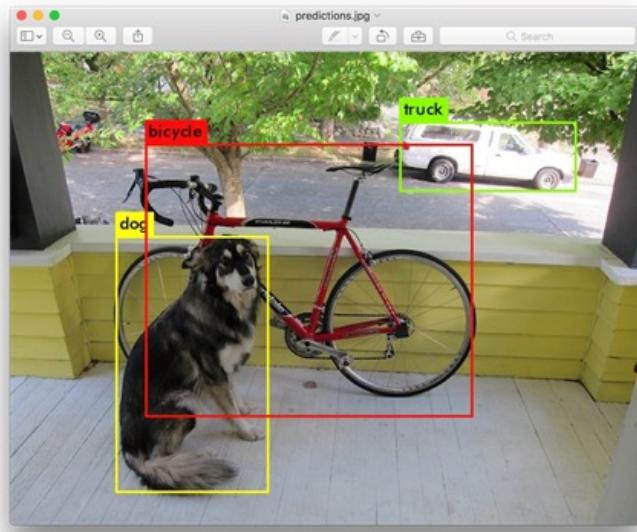


FIGURE A.2 – Image du processus de détection d'objet (citation de YOLOv5)

Prédiction de la boîte englobante (Bounding Box Prediction) YOLOv5 prédit un score d'objectivité pour chaque boîte englobante à l'aide de la régression logistique. Cela devrait être 1 si la boîte englobante antérieure chevauche un objet de vérité terrain de plus que toute autre boîte englobante antérieure. Si la boîte englobante antérieure n'est pas la meilleure mais chevauche un objet de vérité terrain de plus d'un certain seuil, nous ignorons la prédiction. Boîtes englobantes avec priors de dimension et prédiction d'emplacement. Nous prédisons la largeur et la hauteur de la boîte sous forme de décalages par rapport aux centres de gravité du cluster. Nous prédisons les coordonnées du centre de la boîte par rapport à l'emplacement de l'application du filtre à l'aide d'une fonction sigmoïde.

Extracteur de fonctionnalités (Feature Extractor) Darknet-53 Le réseau utilise des couches convolutives successives 3×3 et 1×1 , mais dispose désormais également de connexions de raccourci et est nettement plus grand. Il a 53 couches convolutives.

Darknet-53 est beaucoup plus puissant que Darknet-19, est meilleur que ResNet-101 et 1,5 fois plus rapide et a des performances similaires à ResNet-152 et est 2 fois plus rapide.

Type	Filters	Size	Output
Convolutional	32	3×3	256×256
Convolutional	64	$3 \times 3 / 2$	128×128
1x	Convolutional	32	1×1
1x	Convolutional	64	3×3
	Residual		128×128
	Convolutional	128	$3 \times 3 / 2$
			64×64
2x	Convolutional	64	1×1
2x	Convolutional	128	3×3
	Residual		64×64
	Convolutional	256	$3 \times 3 / 2$
			32×32
8x	Convolutional	128	1×1
8x	Convolutional	256	3×3
	Residual		32×32
	Convolutional	512	$3 \times 3 / 2$
			16×16
8x	Convolutional	256	1×1
8x	Convolutional	512	3×3
	Residual		16×16
	Convolutional	1024	$3 \times 3 / 2$
			8×8
4x	Convolutional	512	1×1
4x	Convolutional	1024	3×3
	Residual		8×8
	Avgpool		Global
	Connected		1000
	Softmax		

FIGURE A.3 – Darknet-53

A.3 Étapes pour construire le modèle

A.3.1 Étapes 1 : L'ensemble de données du projet

L'ensemble de données contient des images aux différents panneaux de signalisation. Il est en outre classé en 3 classes différentes. L'ensemble de données est assez variable, certaines classes ont de nombreuses images tandis que certaines classes ont peu d'images.



FIGURE A.4 – Classes des panneaux de signalisation

A.3.2 Étapes 2 : Convertir les annotations au format YOLOv5

Pour entraîner une modèle sur l'algorithme YOLOv5, nous sommes besoin de créer des fichiers *.txt qui donne la classe, position, et dimension d'objet dans image qu'on veut le détecter, s'écrit comme suit <classe><positionX><positionY><largeur><longueur>.



The screenshot shows a Windows Notepad window titled "7 (35).txt - Bloc-notes". The menu bar includes Fichier, Edition, Format, Affichage, and Aide. The main content area displays the following text:

```
7 0.390118 0.192802 0.181416 0.321290
7 0.625369 0.164619 0.141593 0.253650
7 0.839971 0.195620 0.175516 0.298743
7 0.836283 0.653599 0.132743 0.245195
7 0.611357 0.686010 0.172566 0.310016
7 0.165192 0.687419 0.174041 0.301561
9 0.387168 0.676146 0.134218 0.233921
```

The status bar at the bottom indicates Ln 7, Col 34, 100%, Unix (LF), and UTF-8.

FIGURE A.5 – exemple d'annotation au format YOLO où l'image contient deux objets différents.

A.3.3 Étapes 3 : train le modèle YOLOv5

Dans le package téléchargé de YOLOv5, nous avons 4 versions du modèle : YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l et YOLOv5x. nous allons utiliser le meilleur et c'est YOLOv5x. Vous l'avez deviné, les lettres s, m, l et x correspondent à la taille du modèle. Dans la commande d'entraînement, vous donnez au modèle sélectionné ces arguments :

- img : taille de l'image d'entrée
- batch : taille du lot
- epochs : nombre d'époques d'entraînement
- data : notre fichier yaml créé avant
- cfg : le modèle choisi ici j'ai utilisé le petit
- weights : un chemin personnalisé vers les poids s'il est vide, il sera enregistré dans
- name : noms des résultats
- device :0 pour utiliser le GPU
- cache : images en cache pour une formation plus rapide

3.4 Resultat de training

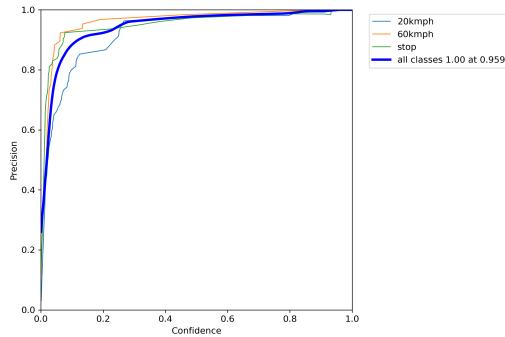


FIGURE 3.6 – Precision

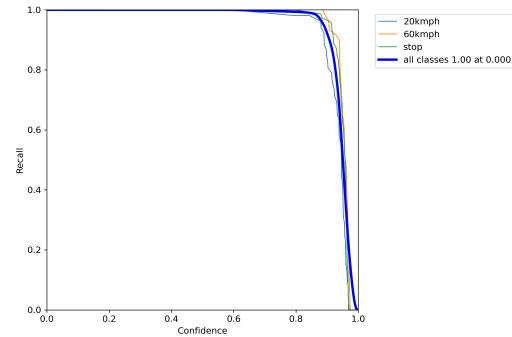


FIGURE 3.7 – Recall

A partir de Matrice de confusion ce modèle est un bon modèle.

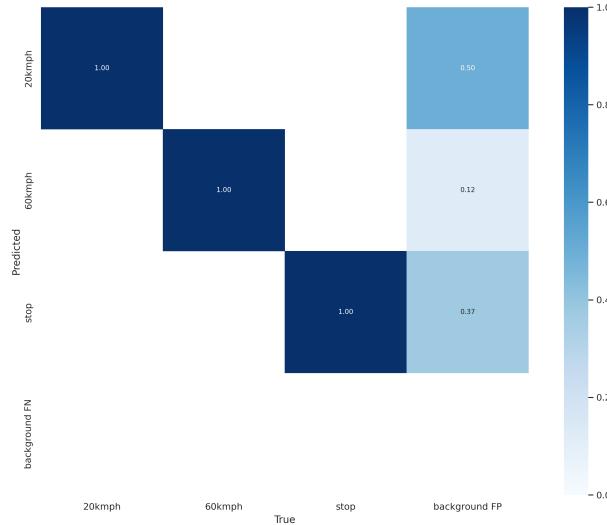


FIGURE 3.8 – Matrice de confision

Les résultats de prédiction et détection des objets.



FIGURE 3.9 – Resultat



FIGURE 3.10 – Resultat

Bibliographie

- [1] Gary Bradski and Adrian Kaehler. *Learning OpenCV : Computer vision with the OpenCV library.* " O'Reilly Media, Inc.", 2008.
- [2] S. Dey. *Image Processing Masterclass with Python : 50+ Solutions and Techniques Solving Complex Digital Image Processing Challenges Using Numpy, Scipy, Pytorch and Keras (English Edition).* BPB PUBN, 2021.
- [3] A Gallagher. Image processing : Principles and applications tinku acharya and ajoy k. ray, authors. *JOURNAL OF ELECTRONIC IMAGING*, 15(3) :039901, 2006.
- [4] Oge Marques. *Practical image and video processing using MATLAB.* John Wiley & Sons, 2011.
- [5] Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.
- [6] Matt Richardson and Shawn Wallace. *Getting started with raspberry PI.* " O'Reilly Media, Inc.", 2012.
- [7] Peter Ritchie. *Practical Microsoft Visual Studio 2015.* Springer, 2016.
- [8] Himanshu Singh. *Practical Machine Learning and Image Processing : For Facial Recognition, Object Detection, and Pattern Recognition Using Python.* Springer, 2019.
- [9] Guido Van Rossum and Fred L Drake Jr. *Python tutorial*, volume 620. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.