



Université Hassan Ier
Faculté des Sciences et Techniques
Département informatique

MASTER : RESEAU ET SYSTÈME INFORMATIQUE

Détection de lignes par OpenCV pour une Voiture Autonome

RÉALISÉ PAR :

LAFRAOUZI Mouhssine

IDRISS AHMED Mohammed

ES-SOUSY Mohamed

SUPERVISÉ PAR:

MR. EZZATI

ANNÉE ACADYMIQUE : 2023/2024

ABSTRAIT

Programme d'études

Informatique- SYSTEME ET RESEAUX

Nom de sujet

Détection de lignes par OpenCV pour une Voiture Autonome

Superviseur EZZATI

La vision par ordinateur est une branche de l'intelligence artificielle qui vise à permettre à une machine de comprendre ce qu'elle "voit" lorsqu'elle est connectée à une ou plusieurs caméras. Avec l'utilisation généralisée des images numériques, les techniques de vision par ordinateur se sont révélées être un outil indispensable dans diverses applications telles que la vidéosurveillance, la compression vidéo, l'imagerie médicale, la robotique, l'interaction homme-machine. Notre projet se concentre sur l'étude d'une méthode de détection des lignes de voie dans les voitures autonomes.

.

Mots clés

Python, Vision par Ordinateur,, Deep Learning, Image, Détection d'objets, Détection de lignes, voiture autonome.

Table des matières

INTRODUCTION GÉNÉRALE :	5
CHAPITRE 01 : VISION PAR ORDINATEUR	8
Introduction :	8
Qu'est-ce qu'une image ?	8
Types d'images :	9
Images binaires :	9
Image en niveaux de gris :	10
Image couleur (RGB) :	10
Caractéristiques d'une image numérique :	11
Pixel :	11
Définition :	12
Dimension :	13
Résolution :	13
Le bruit :	14
Luminance :	14
Contraste :	15
Histogramme :	15
Vision par ordinateur :	16
Qu'est-ce que la vision par ordinateur ?	16
Comment fonctionne la vision par ordinateur :	16
Domaines d'application de la vision par ordinateur :	17
La sécurité :	18
Présentation de la bibliothèque OpenCV :	19
QU'EST-CE QUE LE CV OUVERT ?	19
Principaux modules d'OpenCV 4.2.0 :	20
Conclusion :	21
CHAPITRE 02 : Réalisation :	22
Introduction :	22
Étapes de détection de la ligne de démarcation :	22
Charger une image :	22
Prétraitement d'une image :	23
Détection des contours de Canny :	25
Définir la région d'intérêt :	28

Transformée de Hough :	28
Ajouter les lignes extrapolées à l'image d'entrée :	30
Combiner des segments de ligne en lignes à deux voies :	31
Angle de braquage :	31
Ajouter un pipeline à la vidéo :	32

Tables des figures

Figure 1: Représentation matricielle d'une image numérique	8
Figure 2: Trois types d'images différents	9
Figure 3: Image binaire	10
Figure 4: Niveaux de gris [0-255].	10
Figure 5: COULEUR RVB	11
Figure 6: Représentation d'un pixel dans une image numérique	12
Figure 7: Exemple de définition d'une image	13
Figure 8: Exemple de résolution d'image	14
Figure 9: Exemple d'une image bruitée	14
Figure 10: Exemple de trois valeurs de luminance différentes pour la même image	15
Figure 11: Exemple de trois valeurs de contraste différentes d'une image	15
Figure 12: Exemple d'une image RVB et de son histogramme	16
Figure 13: Comment fonctionne la vision par ordinateur ?	16
Figure 14: Exemple d'utilisation d'opencv dans le domaine des transports	17
Figure 15: L'image montre la détection d'un masque sociale	18
Figure 16: L'image montre la distanciation violation	18
Figure 17: Application de reconnaissance faciale	19
Figure 18: Exemple d'une image d'entrée	23
Figure 19: Conversion des couleurs RVB en niveaux de gris	23
Figure 20: Conversion en une image en niveaux de gris	24
Figure 21: Noyau gaussien et Échantillon Filtre gaussien	24
Figure 22: Ajout d'un flou gaussien à une image en niveaux de gris	25
Figure 23: L'exemple montre le principe du non-maximum	27
Figure 24: L'étape du double seuil	27
Figure 25: L'image originale et son résultat après application du filtre de Canny	28
Figure 26: Application d'un masque sur l'image de sortie Canny	28
Figure 27: Exemple illustrant le fonctionnement de la méthode de la ligne de Hough	30
Figure 28: Application de la transformée de Hough à l'image masquée	30
Figure 29: Représentation des lignes extrapolées sur l'image d'entrée	31
Figure 30: Combinaison de segments de ligne en lignes à deux voies	31
Figure 31: Représentation de l'angle de braquage de la voiture	32
Figure 32: Algorithme de détection de ligne de voie	33

INTRODUCTION GÉNÉRALE :

La vision par ordinateur est une branche de l'intelligence artificielle (IA) qui permet aux ordinateurs et aux systèmes d'extraire des informations significatives d'une seule image ou d'une séquence d'images. Elle vise à automatiser les tâches que le système visuel d'un être humain peut effectuer.

OpenCV (Open-Source Computer Vision) est une bibliothèque d'algorithmes principalement destinés à la vision par ordinateur en temps réel. Avec l'avènement de machines puissantes, nous disposons d'une plus grande puissance de traitement pour travailler. En mettant l'accent sur la vision en temps réel, OpenCV aide les étudiants et les professionnels à mettre en œuvre efficacement des projets et à lancer des recherches en leur fournissant une infrastructure de vision par ordinateur qui n'était auparavant disponible que dans quelques grands laboratoires de recherche.

Une voiture autonome est un véhicule capable de détecter son environnement et de fonctionner sans intervention humaine. Les développeurs de voitures autonomes utilisent des techniques d'intelligence artificielle et travaillent sur de grandes quantités de données qui comprennent des images provenant des caméras des voitures autonomes à partir desquelles la voiture apprend à identifier les feux de circulation, les arbres, les bordures de trottoir, les piétons, les panneaux de signalisation et d'autres éléments de tout environnement de conduite donné.

Dans ce travail, nous nous intéressons à l'écriture et à l'exécution d'un algorithme utilisant la bibliothèque OpenCV pour la détection de lignes.

L'algorithme de détection des lignes est destiné à une voiture autonome afin de maintenir la voiture sur la bonne voie en utilisant la bibliothèque OpenCV.

CHAPITRE 01 : VISION PAR ORDINATEUR

Introduction :

La vision par ordinateur est un domaine de l'intelligence artificielle (IA) qui vise à développer des techniques permettant aux ordinateurs de capturer, d'interpréter, de comprendre et de traiter les objets visuellement perceptibles. Cela permet aux systèmes de vision par ordinateur de réagir de manière appropriée. Dans cette recherche, nous utiliserons OpenCV qui est une bibliothèque de vision artificielle, mais nous devons d'abord comprendre le domaine du traitement d'images et les caractéristiques des images numériques.

Qu'est-ce qu'une image ?

Une image est une représentation plane d'une scène ou d'un objet situé en général dans un environnement tridimensionnel, elle provient du contact des rayons lumineux provenant des objets formant la scène avec un capteur (caméra, scanner, rayons X, etc.). Il s'agit en fait d'une simple représentation spatiale de la lumière. L'image est considérée comme un ensemble de points auxquels est attribuée une grandeur physique (luminance, couleur).

Images numériques :

Une image numérique est une représentation d'une image réelle sous la forme d'un ensemble de nombres qui peuvent être stockés et traités par un ordinateur numérique. Afin de traduire l'image en nombres, elle est divisée en petites zones appelées pixels (éléments d'image). Pour chaque pixel, le dispositif d'imagerie enregistre un nombre, ou un petit ensemble de nombres, qui décrit une propriété de ce pixel, telle que sa luminosité (l'intensité de la lumière) ou sa couleur. Les nombres sont disposés dans un tableau de lignes et de colonnes qui correspondent aux positions verticales et horizontales des pixels dans l'image [1].

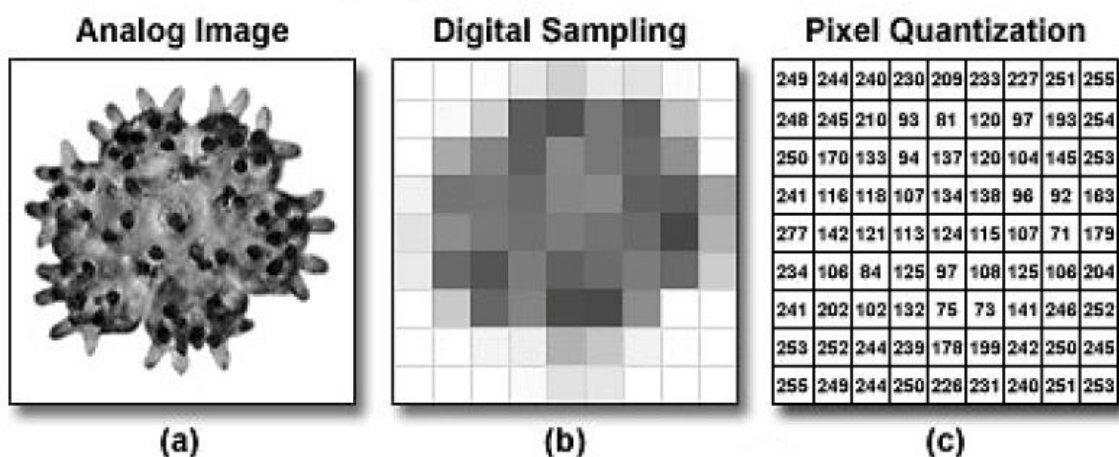


Figure 1: Représentation matricielle d'une image numérique

Types d'images :

Avant d'expliquer la différence entre les trois types d'images, les images binaires, les images en gris et les images en couleur, nous montrons dans la figure ci-dessous un exemple de la même image avec différents types :

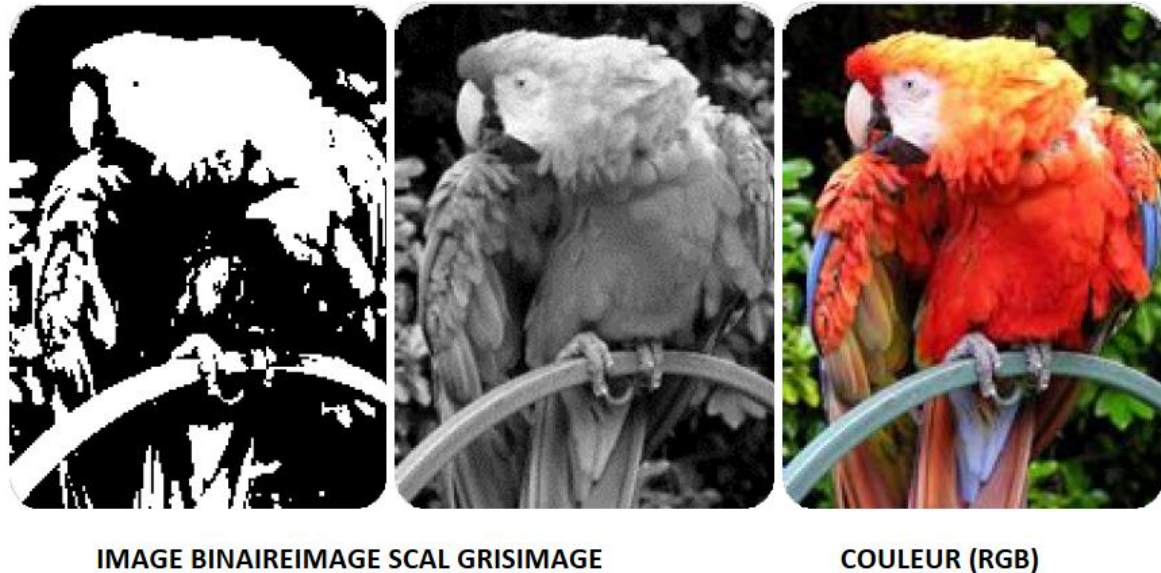


Figure 2: Trois types d'images différents

Images binaires :

Il s'agit du type d'image le plus simple. L'image binaire consiste en une image de 1 bit et il suffit d'un chiffre binaire pour représenter un pixel noir et blanc "0" et "1". Les images binaires sont principalement utilisées pour les formes générales ou les contours.

Les images binaires sont générées en utilisant l'opération de seuil. Lorsqu'un pixel est supérieur à la valeur seuil, il devient blanc ("1") et lorsqu'il est inférieur à la valeur seuil, il devient noir ("0").

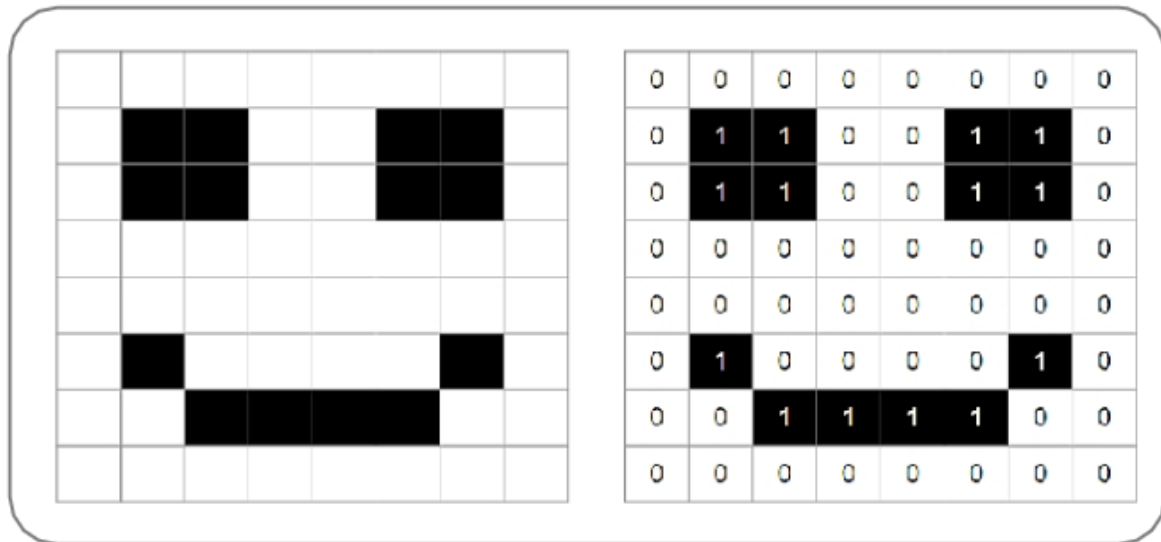


Figure 3:Image binaire

Image en niveaux de gris :

Les images en niveaux de gris sont des images monochromes, c'est-à-dire qu'elles n'ont qu'une seule couleur, les "nuances de gris".

Une image normale en niveaux de gris contient 8 bits/pixel de données, soit 256 niveaux de gris différents.



Figure 4:Niveaux de gris [0-255].

Image couleur (RGB) :

Les images couleur sont des images monochromes à trois bandes dans lesquelles chaque bande contient une couleur différente et l'information réelle est stockée dans l'image numérique. Les images couleur contiennent des informations sur les niveaux de gris dans chaque bande spectrale. Les images sont représentées en rouge, vert et bleu (images RVB), et chaque image couleur comporte 24 bits/pixel, soit 8 bits pour chacune des trois bandes de couleur (RVB) [2].

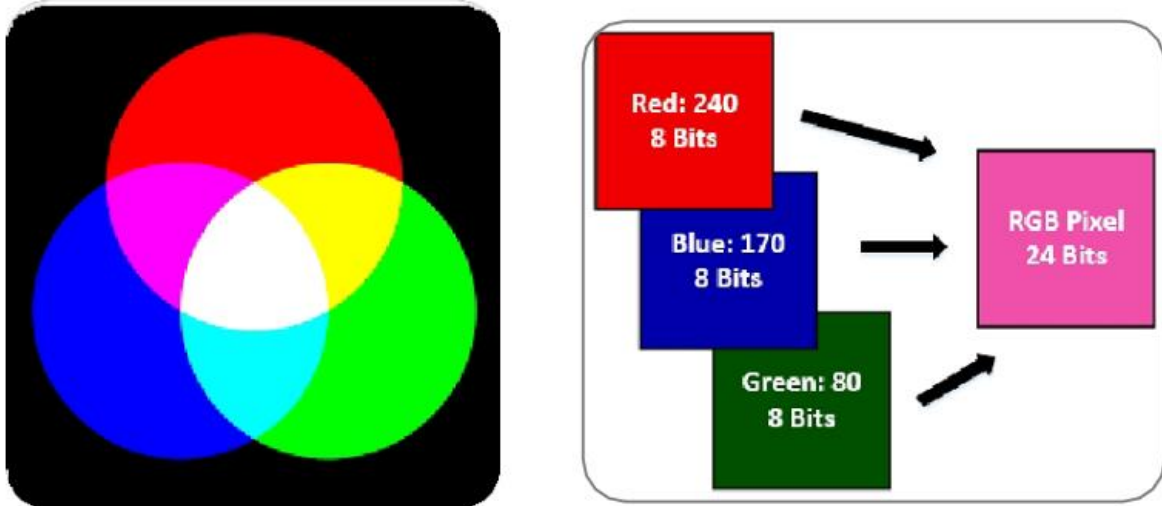


Figure 5: COULEUR RVB

Caractéristiques d'une image numérique :

Pixel :

Une image est constituée d'un ensemble de points appelés "pixel" (Picture element) représentant ainsi le plus petit élément constitutif d'une image numérique. Pour les images en 3D, le "pixel" est appelé voxel et représente un volume élémentaire. On trouve des exemples de telles images dans les images médicales. Les images tomographiques axiales sont ainsi des images construites à partir de plusieurs radiographies prises sous différents angles de vue.

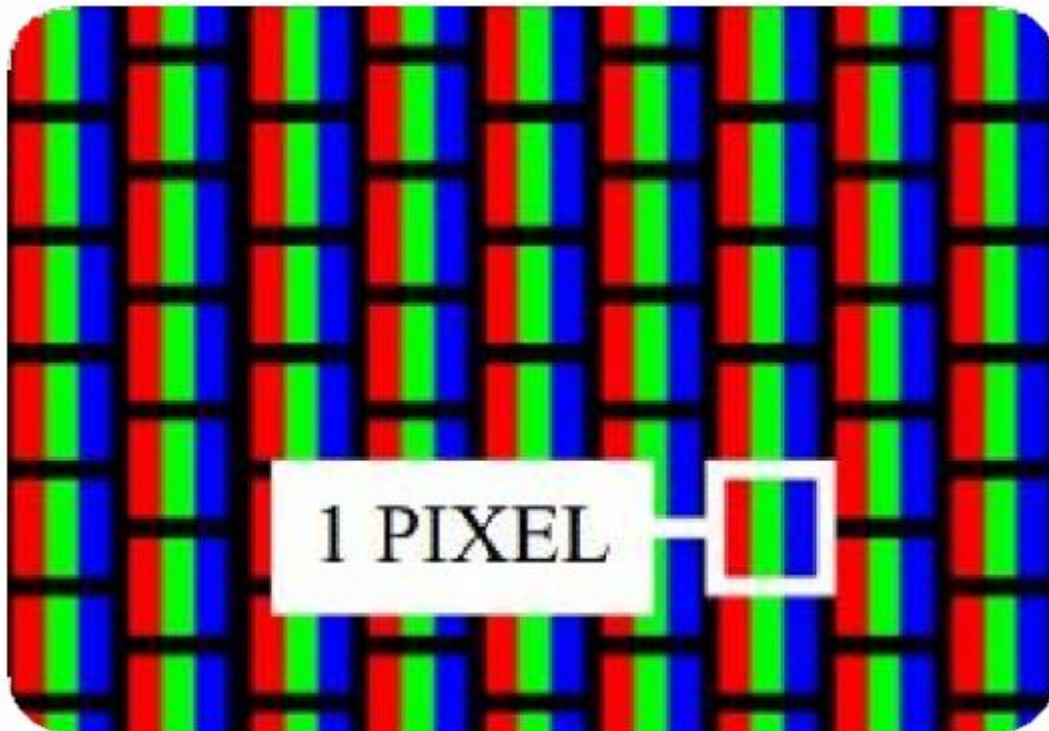


Figure 6: Représentation d'un pixel dans une image numérique

Définition :

Le nombre de points (pixels) constituant une image est appelé définition ; c'est le nombre de colonnes de l'image multiplié par son nombre de lignes. Une image de 10 colonnes et 11 lignes aura une définition de 10×11 , soit 110 pixels.

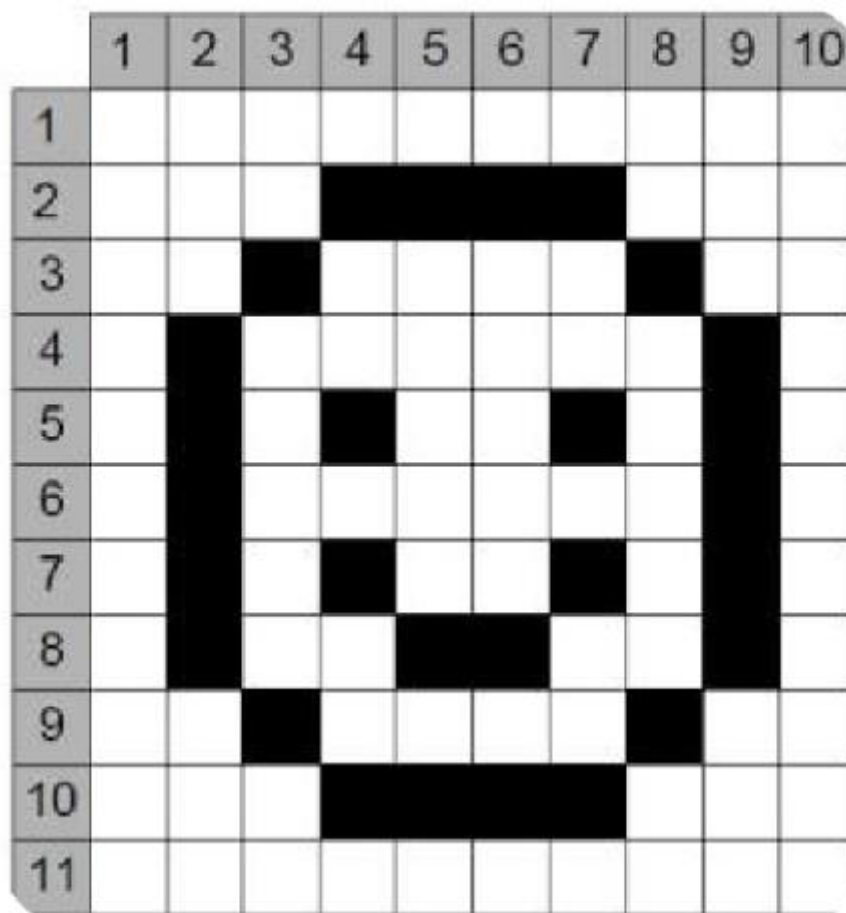


Figure 7: Exemple de définition d'une image

Dimension :

La dimension d'une image correspond aux mesures (*width x length*) d'une image numérique, par exemple 4608x3072 px. En général, on commence par indiquer la largeur. La dimension d'un fichier graphique est donnée en pixels.

Résolution :

La résolution d'une image est le nombre de points contenus dans une longueur donnée (en pouces).

Elle s'exprime en points par pouce (DPI en français ou DPI en anglais pour Dots Per Inch).

Un pouce mesure 2,54 cm, c'est une unité de mesure britannique utilisée dans les pays anglophones.

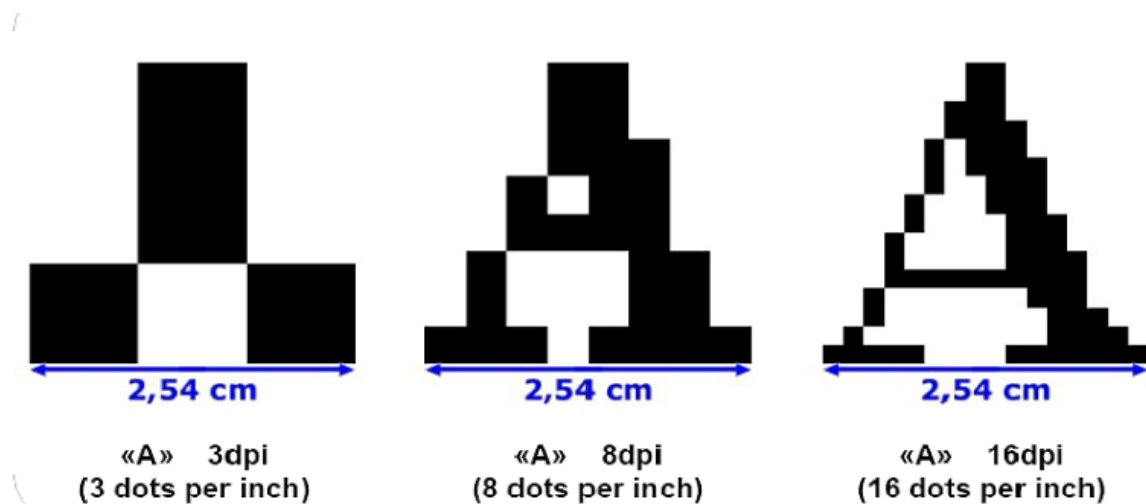


Figure 8: Exemple de résolution d'image

Le bruit :

Le bruit est une variation aléatoire de la luminosité ou des informations de couleur dans les images, et il s'agit généralement d'un aspect du bruit électronique. Il peut être produit par le capteur d'image et les circuits d'un scanner ou d'un appareil photo numérique. Le bruit d'image peut également provenir du grain de la pellicule et de l'image.

le bruit de fond inévitable d'un détecteur de photons idéal. Le bruit d'image est un sous-produit indésirable de la capture d'image qui obscurcit les informations souhaitées.



Image normale



bruyante

Figure 9: Exemple d'une image bruitée

Luminance :

La luminance est l'intensité lumineuse projetée sur une surface et dans une direction données. La luminance décrit généralement l'intensité de la lumière émise. Dans le cas de nos

produits de profilage d'écran, nous mesurons la luminance en cd/m^2 comme unité de mesure.



Faible luminance

Luminance normale

Haute luminance

Figure 10: Exemple de trois valeurs de luminance différentes pour la même image

Contraste :

Le contraste est l'échelle de différence entre le noir pur et le blanc pur. Dans la plupart des cas, lorsque vous sortez votre téléphone et prenez un selfie, vous pouvez obtenir une image à contraste moyen. Si vous souhaitez faire ressortir l'objet, vous pouvez utiliser la lumière et l'ombre pour augmenter ou diminuer le contraste



Faible contraste

Contraste normal

Contraste élevé

Figure 11: Exemple de trois valeurs de contraste différentes d'une image

Histogramme :

Un histogramme d'image est un graphique de l'intensité des pixels (sur l'axe des x) en fonction du nombre de pixels (sur l'axe des y). L'axe des x présente tous les niveaux de gris disponibles et l'axe des y indique le nombre de pixels qui ont une valeur de niveau de gris particulière.² Plusieurs niveaux de gris peuvent être combinés en groupes afin de réduire le nombre de valeurs individuelles sur l'axe des x.



Image originale

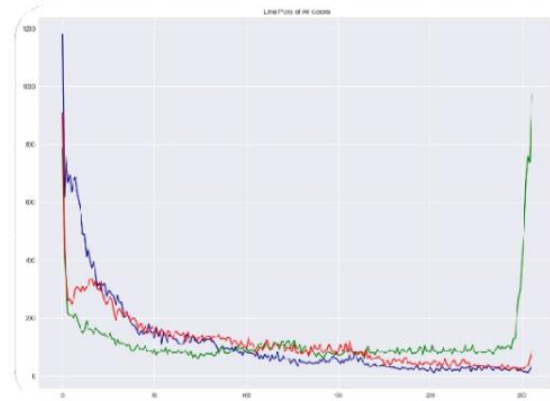


Figure 12: Exemple d'une image RVB et de son histogramme

Vision par ordinateur :

Qu'est-ce que la vision par ordinateur ?

La vision par ordinateur est un ensemble d'algorithmes qui permettent à un ordinateur d'analyser une image et d'en extraire des informations utiles. Elle est utilisée dans de nombreuses applications et devient rapidement un élément de la vie quotidienne.

Comment fonctionne la vision par ordinateur :

Tout commence par une image. L'ordinateur analyse une image pour identifier les lignes, les coins et une large palette de couleurs. Une fois les caractéristiques extraites, l'ordinateur peut utiliser ces informations pour de nombreuses tâches différentes.

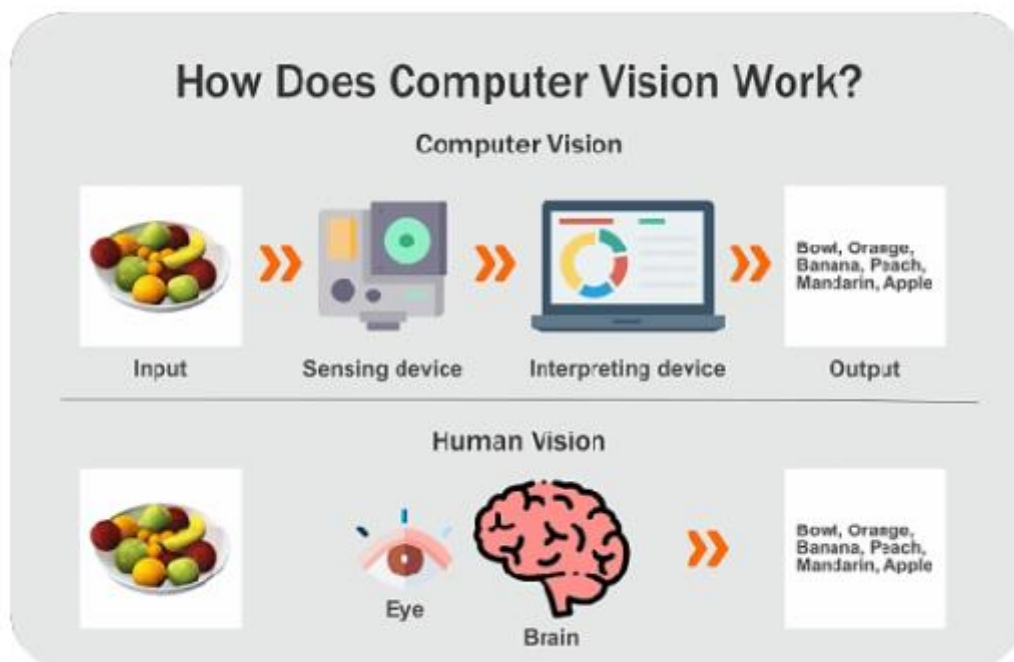


Figure 13: Comment fonctionne la vision par ordinateur ?

Domaines d'application de la vision par ordinateur :

La vision par ordinateur est une fonction puissante qui peut être combinée avec de nombreux types d'applications et de dispositifs de détection pour soutenir un certain nombre de cas d'utilisation pratiques. Voici quelques-uns des différents types de champs d'application de la vision par ordinateur .

Transport :

Les exigences croissantes du secteur des transports ont propulsé le développement technologique dans cette industrie, avec la vision par ordinateur en son centre. Par exemple, les voitures à conduite autonome utilisent l'identification et le suivi d'objets en temps réel pour recueillir des informations sur ce qui se passe autour d'elles et se diriger en conséquence.



Figure 14:Exemple d'utilisation d'opencv dans le domaine des transports

Sécurité / COVID-19 Mesures de protection en cas de pandémie :

Lorsque la pandémie de Covid-19 a frappé le monde au début de l'année 2020, elle a entraîné l'arrêt de plusieurs activités manufacturières. Même lorsque les activités ont repris, plusieurs usines de fabrication ont rendu obligatoires la distanciation sociale et le port de masques pour la sécurité des travailleurs. Les applications de vision par ordinateur sont extrêmement utiles dans ce cas, car elles permettent de surveiller efficacement la main-d'œuvre afin d'identifier toute violation du protocole Covid-19. En outre, l'utilisation de ces modèles de vision par ordinateur dans les usines de fabrication garantira un environnement de travail sûr pendant et même après la pandémie.



Figure 15:L'image montre la détection d'un masque sociale



Figure 16:L'image montre la distanciation violation

La sécurité :

Les caméras de reconnaissance faciale font partie intégrante de nombreuses villes intelligentes. Ces caméras surveillent la ville à la recherche d'activités criminelles, reconnaissent les objets en mouvement et détectent les actes répréhensibles. En outre, les

systèmes de surveillance intelligents peuvent envoyer des alertes en cas d'activité irrégulière dans la ville. Les signaux sont envoyés avec l'aide des agences de sécurité publique.



Figure 17: Application de reconnaissance faciale

Présentation de la bibliothèque OpenCV :

QU'EST-CE QUE LE CV OUVERT ?

OpenCV (également connu sous le nom de Open-Source Computer Vision) est une bibliothèque open-source pour la vision par ordinateur et l'apprentissage automatique. Elle offre de nombreuses fonctionnalités pour le traitement d'images et la vision par ordinateur. Il s'agit d'une bibliothèque multiplateforme, qui fonctionne avec de nombreux langages de programmation et systèmes d'exploitation. La bibliothèque contient plus de 2 500 algorithmes optimisés pour les tâches d'apprentissage automatique et de vision par ordinateur. Elle dispose d'une communauté de plus de 47 000 professionnels de la vision par ordinateur et a été téléchargée plus de 18 millions de fois.



Principaux modules d'OpenCV 4.2.0 :

OpenCV se compose de deux types de modules, les modules principaux et les modules supplémentaires :

- **Modules principaux** : Ces modules sont plus ou moins les modules de base d'OpenCV et sont livrés par défaut avec les versions packagées. Ils constituent de modules centraux car ils fournissent les fonctionnalités de base telles que les tâches de traitement d'images, le filtrage, la transformation, etc.

Modules supplémentaires : Ces modules ne sont pas fournis par défaut avec la distribution OpenCV. Ces modules sont liés à des fonctionnalités supplémentaires de vision par ordinateur telles que la reconnaissance de texte.

Modules principaux :

a) Cœur de métier :

Comprend toutes les fonctionnalités de base d'OpenCV, telles que les structures de base, les classes Mat, etc.

b) Imgproc :

Inclut des fonctions de traitement d'images telles que les transformations, les manipulations, le filtrage, etc.

c) Imgcodecs :

Comprend des fonctions de lecture et d'écriture d'images.

d) Videoio :

Comprend des fonctions de lecture et d'écriture de vidéos.

e) highgui :

Inclut des fonctions pour la création d'une interface graphique permettant de visualiser les résultats.

f) Vidéo :

Il comprend des fonctions d'analyse vidéo telles que la détection et le suivi des mouvements, le filtre de Kalman et le fameux algorithme CaM Shift (utilisé pour le suivi des objets).

g) calib3d :

Inclut des fonctions d'étalonnage et de reconstruction 3D utilisées pour l'estimation de la transformation entre deux images.

h) caractéristiques2d :

Inclut des fonctions pour les algorithmes de détection de points clés et d'extraction de descripteurs qui sont utilisés dans les algorithmes de détection et de catégorisation d'objets.

i) Objdetect :

Prise en charge de la détection d'objets.

j) dnn :

Il est notamment utilisé pour la détection et la classification d'objets. Le module dnn est relativement nouveau dans la liste des modules principaux et prend en charge l'apprentissage profond.

k) ml :

Il comprend des fonctions de classification et de régression et couvre la plupart des capacités d'apprentissage automatique.

l) Flann :

Prend en charge des algorithmes optimisés qui traitent de la recherche du plus proche voisin pour des caractéristiques de haute dimension dans de grands ensembles de données. Flann est l'acronyme de Fast Library for approximate nearest neighbors (FLann).

m) Photo :

Comprend des fonctions de vision par ordinateur liées à la photographie, telles que la suppression du bruit, la création d'images HD, etc.

n) Couture :

Inclut des fonctions d'assemblage d'images qui font appel à des concepts tels que la rotation, l'estimation et la déformation d'images.

o) Forme :

Inclut des fonctions qui traitent de la transformation des formes, de l'appariement et des sujets liés à la distance.

p) Superres :

Comprend des algorithmes qui gèrent la résolution et l'amélioration.

q) Vidéostab :

Inclut les algorithmes utilisés pour la stabilisation vidéo.

r) Viz :

Afficher les widgets dans une fenêtre de visualisation 3D.

Conclusion :

Dans ce chapitre, nous avons introduit quelques notions de base sur les images, qui seront nos données d'entrée, et nous avons donné quelques détails sur le domaine de la vision par ordinateur pour comprendre pourquoi nous utiliserons OpenCV, qui est une bibliothèque de vision par ordinateur, de sorte que nous avons au moins compris ce qu'est la vision par ordinateur et ses applications. Et pour conclure, nous avons donné la structure de la bibliothèque OpenCV.

CHAPITRE 02 : Réalisation :

Introduction :

Une voiture auto-conduite, également appelée véhicule autonome, est une voiture qui détecte son environnement et se déplace en toute sécurité avec peu ou pas d'interaction humaine. L'identification des lignes de la voie est une tâche très importante pour maintenir le véhicule dans les contraintes de la voie. Dans notre projet, nous allons décrire un pipeline simple pour la détection des voies et l'implémenter sur un Raspberry Pi 4 que nous monterons sur une petite voiture robotisée, puis nous placerons cette dernière sur un circuit pour voir comment elle se comporte. Mais avant de pouvoir détecter des lignes de voie dans une vidéo, nous devons être capables de détecter des lignes de voie dans une seule image. Une fois cette étape franchie, il suffit de répéter les mêmes étapes pour toutes les images d'une vidéo. Pour faciliter cette tâche, nous utiliserons la bibliothèque OpenCV-Python.

Étapes de détection de la ligne de démarcation :

Pour atteindre notre objectif, nous suivrons les étapes suivantes :

1. Charger une image.
2. Prétraitement de l'image :
 - 2.1. Convertir en niveaux de gris.
 - 2.2. Flou gaussien.
3. Détection des contours de Canny.
4. Région d'intérêt.
5. Transformée de Hough.
6. Ajouter les lignes extrapolées à l'image d'entrée.
7. Combiner des segments de ligne en lignes à deux voies.
8. Angle de braquage.
9. Ajouter un pipeline à la vidéo.

Ensuite, nous expliquons comment nous mettons en œuvre chaque étape à l'aide d'OpenCV.

Charger une image :

Pour lire une image dans OpenCV, nous utilisons la fonction *imread* :

Syntaxe : `cv2.imread ("Path", Flag)`.



Figure 18:Exemple d'une image d'entrée

Prétraitement d'une image :

1. Convertir l'image originale en niveaux de gris :

Nous prenons comme exemple l'image ci-dessous et appliquons toutes les étapes de détection sur celle-ci. Notre objectif est de trouver les voies jaunes et blanches. Pour ce faire, nous devons tout d'abord convertir l'image originale en niveaux de gris. La moyenne pondérée des niveaux de gris est représentée par l'équation suivante :

Niveaux de gris = $0,299 * R + 0,587 * V + 0,114 * B$

Où R, G et B sont des nombres entiers représentant le rouge (R), le vert (G) et le bleu (B), avec des valeurs comprises entre 0 et 255 .







	Pure Red (255,0,0)		Equivalent Gray (76, 76, 76)
	Pure Green (0, 255, 0)		Equivalent Gray (150, 150, 150)
	Pure Blue (0, 0, 255)		Equivalent Gray (29, 29, 29)

Figure 19:Conversion des couleurs RVB en niveaux de gris

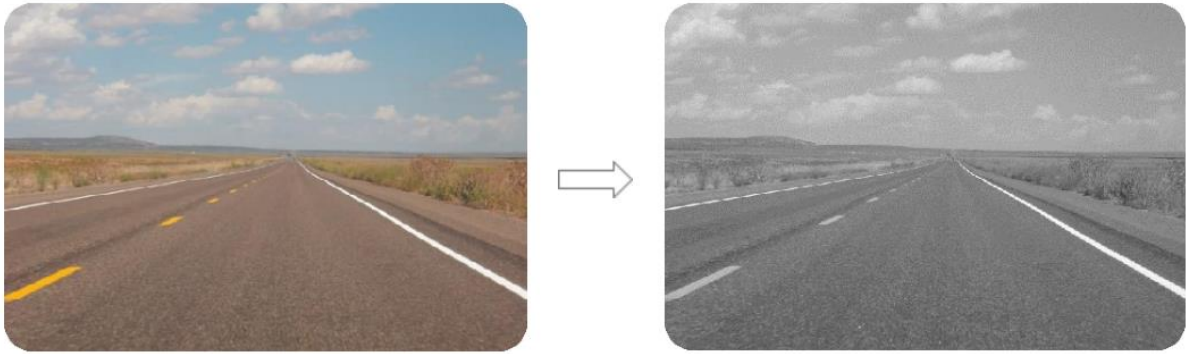


Figure 20: Conversion en une image en niveaux de gris

2. Appliquez un léger flou gaussien :

Dans les systèmes réels, les images capturées sont bruitées et il est donc nécessaire de les filtrer avant de détecter les bords afin d'éviter la détection de faux contours.

Le flou gaussien (lissage gaussien) est une étape de prétraitement utilisée pour réduire le bruit de l'image. Ce filtre fonctionne en prenant un pixel et en calculant une valeur (similaire à la moyenne, mais avec plus de biais au milieu). Le filtre est construit sur la base d'une distribution normale, qui a la forme d'une courbe en cloche. L'idée est que les pixels les plus proches du pixel central ont un poids plus important que ceux qui en sont plus éloignés.

Pour plus de clarté, voici un exemple de noyau gaussien appliqué à une petite matrice.

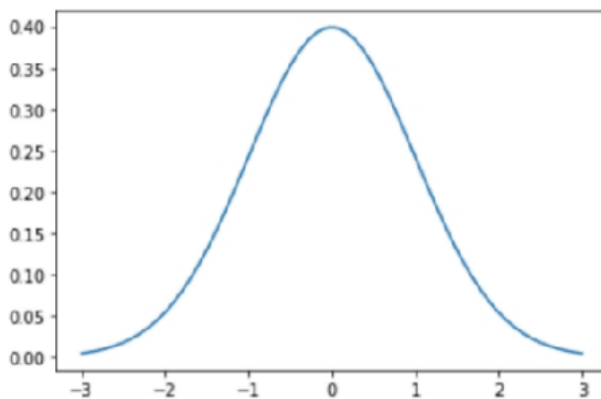


Figure 21: Noyau gaussien et Échantillon Filtre gaussien

Nous appliquons le filtre à la matrice ci-dessous.

100	100	1 00	2 00	1 00
100	100	1 00	4 00	2 00
100	100	1 00	2 00	1 00
100	100	100	100	100
100	100	100	100	100

Pour, pour modifier l'pixel(2,4) :

$4 \times 200 + 2 \times (200 + 200 + 100 + 100) + 1 \times (200 + 200 + 200 + 100) = 2700$.

$2700 / (4 + 2 \times 4 + 1 \times 4) = 168.75$.

Après avoir appliqué le filtre sur ce segment, la couleur se situera entre le bleu et le rose, mais plutôt du côté du rose. Cela créera probablement un effet de dégradé et atténuera les bords abrupts.

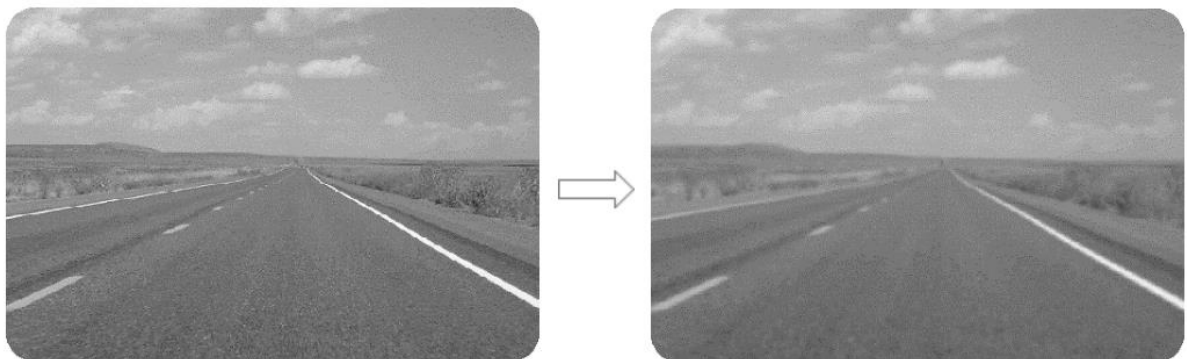


Figure 22: Ajout d'un flou gaussien à une image en niveaux de gris

Pour ce faire dans OpenCV, nous appliquons simplement cette fonction : **cv2. GaussianBlur()**.

Détection des contours de Canny :

Les conditions complexes de la route rendent très difficile la détection correcte des bords des marquages de voies. Afin d'obtenir un bord idéal des marquages de voies dans une

image de route, nous appliquons le détecteur de Canny, un opérateur de détection de bord qui utilise un algorithme en plusieurs étapes pour détecter une large gamme de bords dans les images.

- **Calcul du gradient :**

L'étape de calcul du gradient permet de détecter l'intensité et la direction des contours en calculant le gradient de l'image à l'aide d'opérateurs de détection des contours. Les gradients peuvent être déterminés en utilisant des filtres de Sobel pour les deux directions (horizontale et verticale) où A est l'image. Un bord se produit lorsque la couleur d'une image change et que l'intensité du pixel change également.

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * A ; G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * A$$

Nous calculons ensuite l'ampleur et l'angle des gradients directionnels :

Magnitude : $|G| = \sqrt{G_x^2 + G_y^2}$

Angle : $\theta = \tan^{-1} (G_y/G_x)$

- **Suppression non maximale :**

Idéalement, l'image finale devrait avoir des bords fins. Nous devons donc procéder à une suppression non maximale pour amincir les bords en balayant l'ensemble de l'image afin de nous débarrasser des pixels qui pourraient ne pas faire partie d'un bord. La suppression des non-maximums consiste à trouver les pixels qui sont des maximums locaux dans la direction du gradient (la direction du gradient est perpendiculaire aux bords).

Par exemple, dans la figure 28, nous avons trois pixels qui se trouvent l'un à côté de l'autre : les pixels a, b et c. L'intensité du pixel b est supérieure à celle de a et de c, les pixels a et c se trouvant dans la direction du gradient de b. Par conséquent, le pixel b est considéré comme une arête. Dans le cas contraire, si le pixel b n'était pas un maximum local, il serait fixé à 0 (c'est-à-dire noir), ce qui signifierait qu'il ne s'agirait pas d'un pixel d'arête.

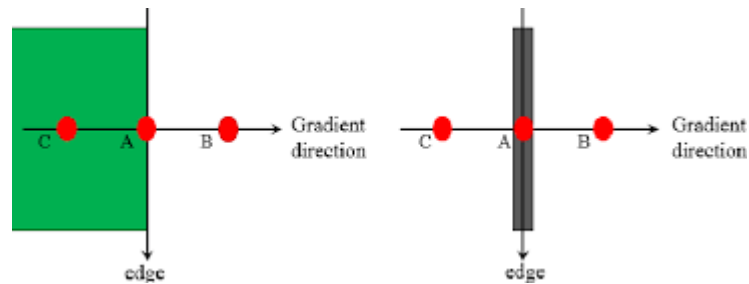


Figure 23: L'exemple montre le principe du non-maximum.

- **Seuil :**

La suppression non maximale n'est pas parfaite car certaines arêtes peuvent être du bruit et non de vraies arêtes. Pour résoudre ce problème, le détecteur de contours de Canny va plus loin et applique un seuillage pour supprimer les contours les plus faibles et conserver les plus forts. Les pixels d'arête qui sont à la limite de la faiblesse ou de la force ne sont considérés comme forts que s'ils sont connectés à des pixels d'arête forts.

Le seuillage définit deux seuils, un seuil haut et un seuil bas. Dans cet algorithme, nous avons normalisé toutes les valeurs de manière à ce qu'elles soient comprises entre 0 et 1.

Les pixels ayant une valeur élevée sont plus susceptibles d'être des bords. Par exemple, vous pouvez choisir un seuil élevé de 0,7, ce qui signifie que tous les pixels dont la valeur est supérieure à 0,7 seront des arêtes fortes. Vous pouvez également choisir un seuil bas de 0,3, ce qui signifie que tous les pixels dont la valeur est inférieure à ce seuil ne sont pas des arêtes, et le fixer à 0. Les valeurs comprises entre 0,3 et 0,7 sont des arêtes faibles.

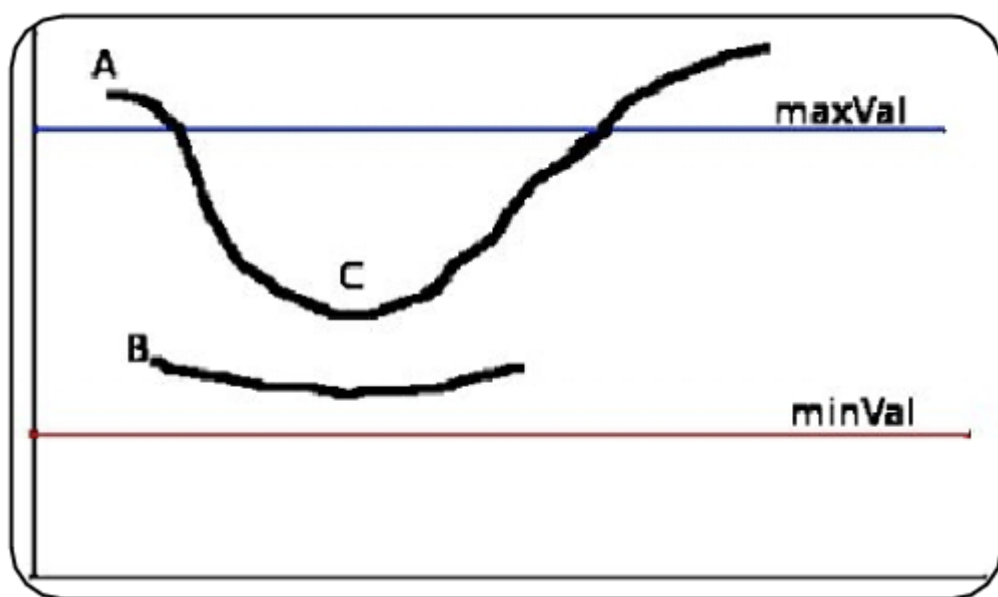


Figure 24: L'étape du double seuil

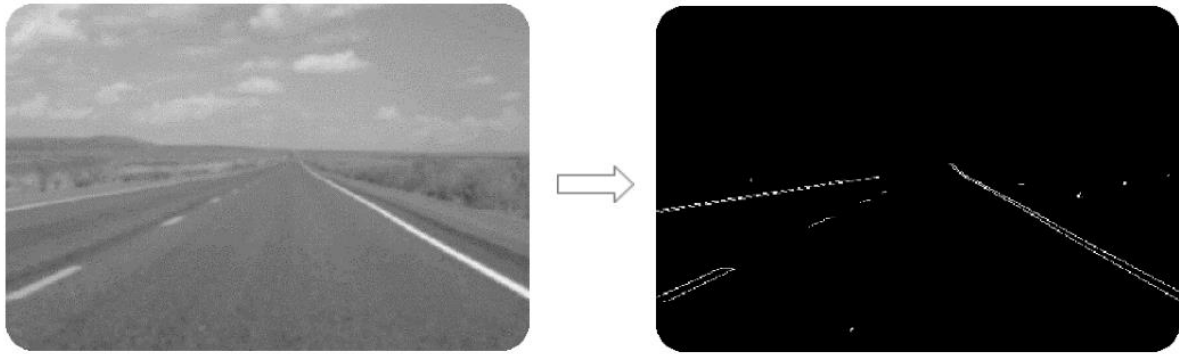


Figure 25: L'image originale et son résultat après application du filtre de Canny

Pour ce faire dans OpenCV, nous appliquons simplement cette fonction : ***cv2. Canny()***.

Définir la région d'intérêt :

La région d'intérêt de la caméra de la voiture est uniquement constituée des deux voies se trouvant immédiatement dans son champ de vision, à l'exclusion de tout autre élément. Nous pouvons filtrer les pixels étrangers en créant une région d'intérêt polygonale et en supprimant tous les autres pixels qui ne se trouvent pas dans le polygone.

Lorsque l'objet Masque est ajouté à la zone d'image, seule la zone non nulle est visible, et toutes les valeurs de pixels du Masque qui se superposent à l'image sont invisibles. En d'autres termes, la forme et la taille de la zone du masque déterminent directement ce que vous voyez. La taille et la forme de l'image finale.

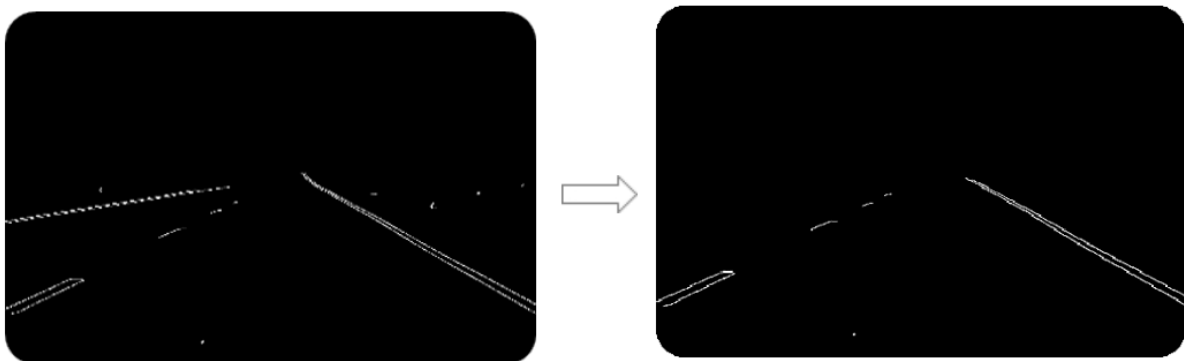


Figure 26: Application d'un masque sur l'image de sortie Canny

Transformée de Hough :

Maintenant que nous avons détecté les bords de la région d'intérêt, nous voulons identifier les lignes, qui indiquent les voies de circulation. C'est là que la transformée de Hough s'avère utile. Avant d'expliquer le fonctionnement de la transformée de Hough, rappelons qu'une ligne droite peut être représentée par deux paramètres.

1. La ligne est alors décrite comme suit : $y = a x + b$, (a, b) correspondent à la pente et à

l'ordonnée à l'origine.

2. ou $r = x \cos \theta + y \sin \theta$, r , est la distance la plus courte entre l'origine et la ligne (s'approchant de la ligne perpendiculairement). Le second, θ , est l'angle entre l'axe des x et la ligne de distance.

- **Comment fonctionne la méthode de la ligne de Hough**

Tout d'abord, il crée un tableau 2D ou accumulateur (pour contenir les valeurs de deux paramètres) qui est initialement fixé à zéro. Les lignes représentent le r et les colonnes le (θ) thêta. La taille du tableau dépend ensuite de la précision dont vous avez besoin. Si la précision des angles doit être de 1 degré, vous avez besoin de 180 colonnes. Pour " r ", si vous avez besoin d'une précision d'un pixel, vous prenez la distance maximale possible, qui est la longueur de la diagonale de l'image. L'exemple ci-dessous explique mieux l'idée sous-jacente.

Considérons une image de 100×100 avec une ligne horizontale au milieu. Prenez le premier point de la ligne. Vous connaissez ses valeurs (x, y) . Maintenant, dans l'équation de la droite, mettez les valeurs $\theta(\text{theta}) = 0, 1, 2, \dots, 180$ et vérifiez le r que vous obtenez. Pour chaque paire $(r, 0)$ (r est mesuré en pixels et 0 est mesuré en radians), vous incrémentez la valeur d'une unité dans l'accumulateur dans sa section les cellules correspondantes $(r, 0)$. Ainsi, dans l'accumulateur, la cellule $(50, 90) = 1$ avec quelques les autres cellules. Prenez maintenant le deuxième point de la ligne. Faites la même chose que précédemment. Incrémentez les valeurs des cellules correspondant à $(r, 0)$ que vous avez obtenues. Cette fois, la cellule $(50, 90) = 2$. Nous sommes en fait voter les valeurs $(r, 0)$. Vous continuez ce processus pour chaque point de la ligne. À chaque point, la cellule $(50, 90)$ sera incrémentée ou votée, tandis que les autres cellules pourront ou non être votées. Ainsi, à la fin, la cellule $(50, 90)$ aura le maximum de votes. Par conséquent, si vous recherchez le maximum de votes dans l'accumulateur, vous obtenez la valeur $(50, 90)$ qui indique qu'il y a une ligne dans cette image à la distance 50 du point d'origine et à un angle de 90 degrés.

Pour ce faire dans OpenCV, nous appliquons simplement cette fonction : ***cv2. HoughLines()***.

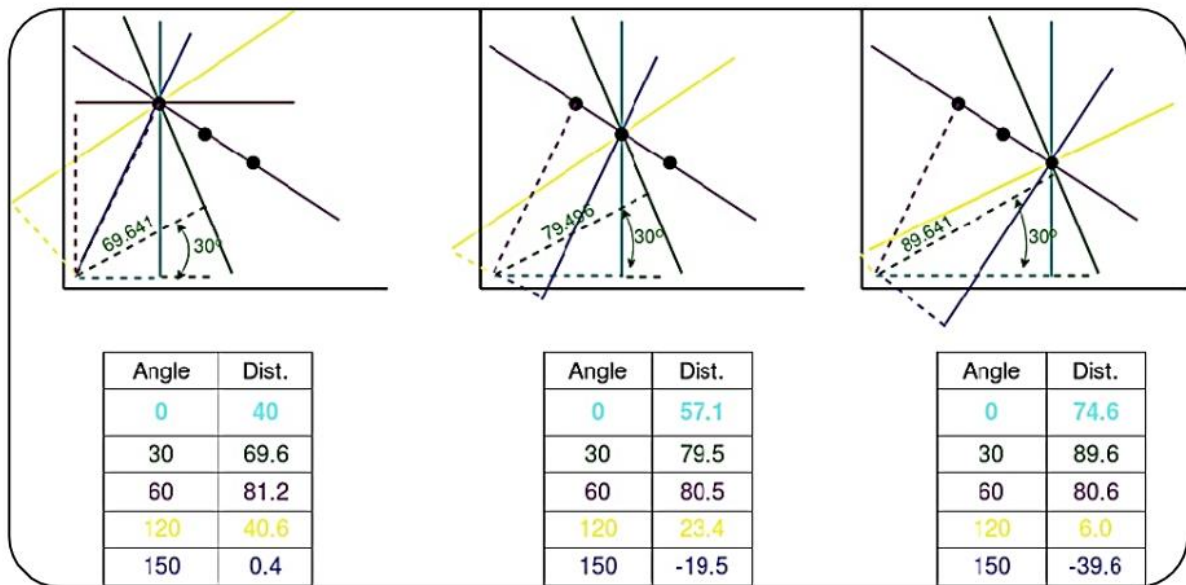


Figure 27: Exemple illustrant le fonctionnement de la méthode de la ligne de Hough

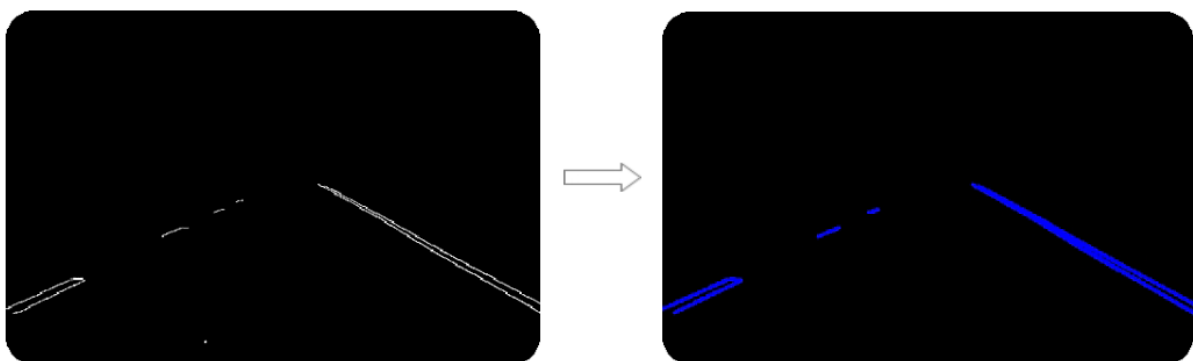


Figure 28: Application de la transformée de Hough à l'image masquée

Ajouter les lignes extrapolées à l'image d'entrée :

Nous superposons ensuite les lignes extrapolées à l'image d'entrée. Pour ce faire, nous ajoutons une valeur de pondération à l'image originale en fonction des coordonnées des lignes détectées.



Figure 29: Représentation des lignes extrapolées sur l'image d'entrée

Combiner des segments de ligne en lignes à deux voies :

Maintenant que nous disposons de nombreux petits segments de droite dont les coordonnées des extrémités sont $(x1, y1)$ et $(x2, y2)$, comment pouvons-nous les combiner pour obtenir les deux lignes qui nous intéressent vraiment, à savoir les lignes de gauche et de droite ? L'une des solutions consiste à classer ces segments de ligne en fonction de leur pente. L'image ci-dessus montre que tous les segments de droite appartenant à la ligne de gauche doivent être inclinés vers le haut et se trouver du côté gauche de l'écran, tandis que tous les segments de droite appartenant à la ligne de droite doivent être inclinés vers le bas et se trouver du côté droit de l'écran. Une fois les segments de ligne classés en deux groupes, il suffit de prendre la moyenne des pentes et des ordonnées à l'origine des segments de ligne pour obtenir les pentes et les ordonnées à l'origine des lignes de voie gauche et droite.

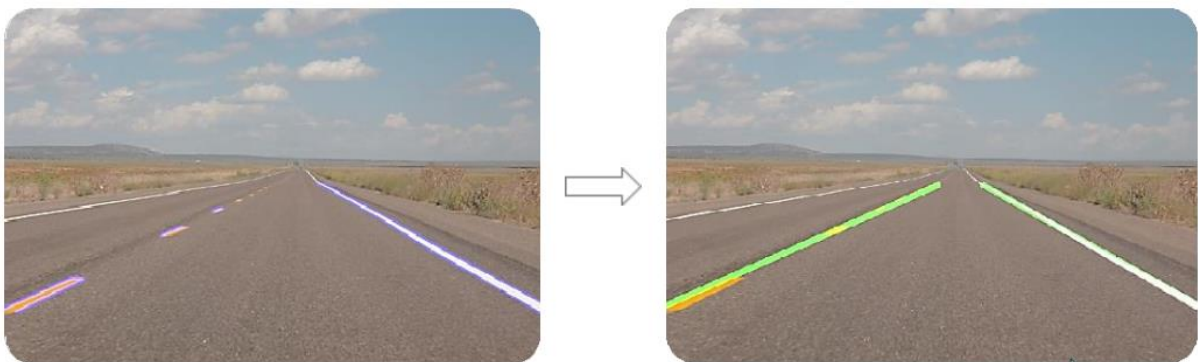


Figure 30: Combinaison de segments de ligne en lignes à deux voies

Angle de braquage :

Maintenant que nous avons les coordonnées des lignes de la voie, nous devons diriger la voiture de manière à ce qu'elle reste à l'intérieur des lignes de la voie, mieux encore, nous devrions essayer de la maintenir au milieu de la voie. Nous devons calculer l'angle de braquage de la voiture, compte tenu des lignes de démarcation détectées. Dans la figure ci-

dessous, la ligne rouge au milieu représente l'angle de braquage de la voiture.

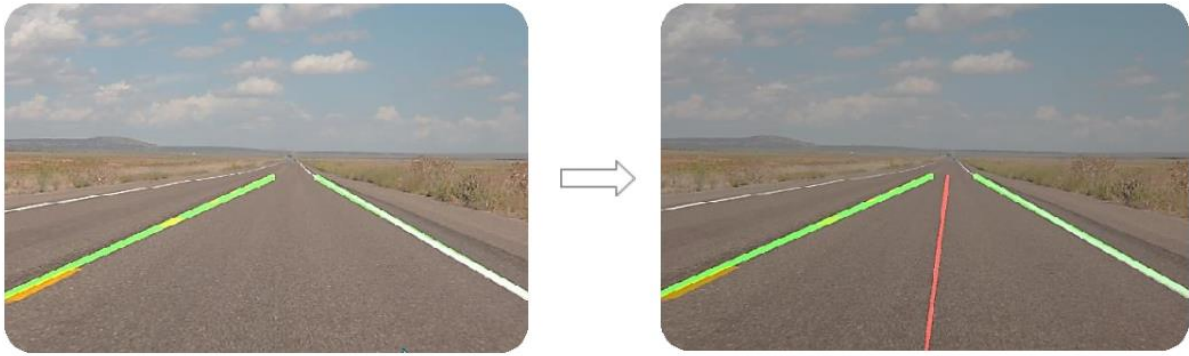


Figure 31: Représentation de l'angle de braquage de la voiture

Ajouter un pipeline à la vidéo :

Une fois que nous avons conçu et mis en œuvre l'ensemble du pipeline de détection des voies de circulation, nous appliquons la transformation image par image. Pour ce faire, nous avons besoin d'une vidéo d'une voiture circulant sur les voies de circulation.

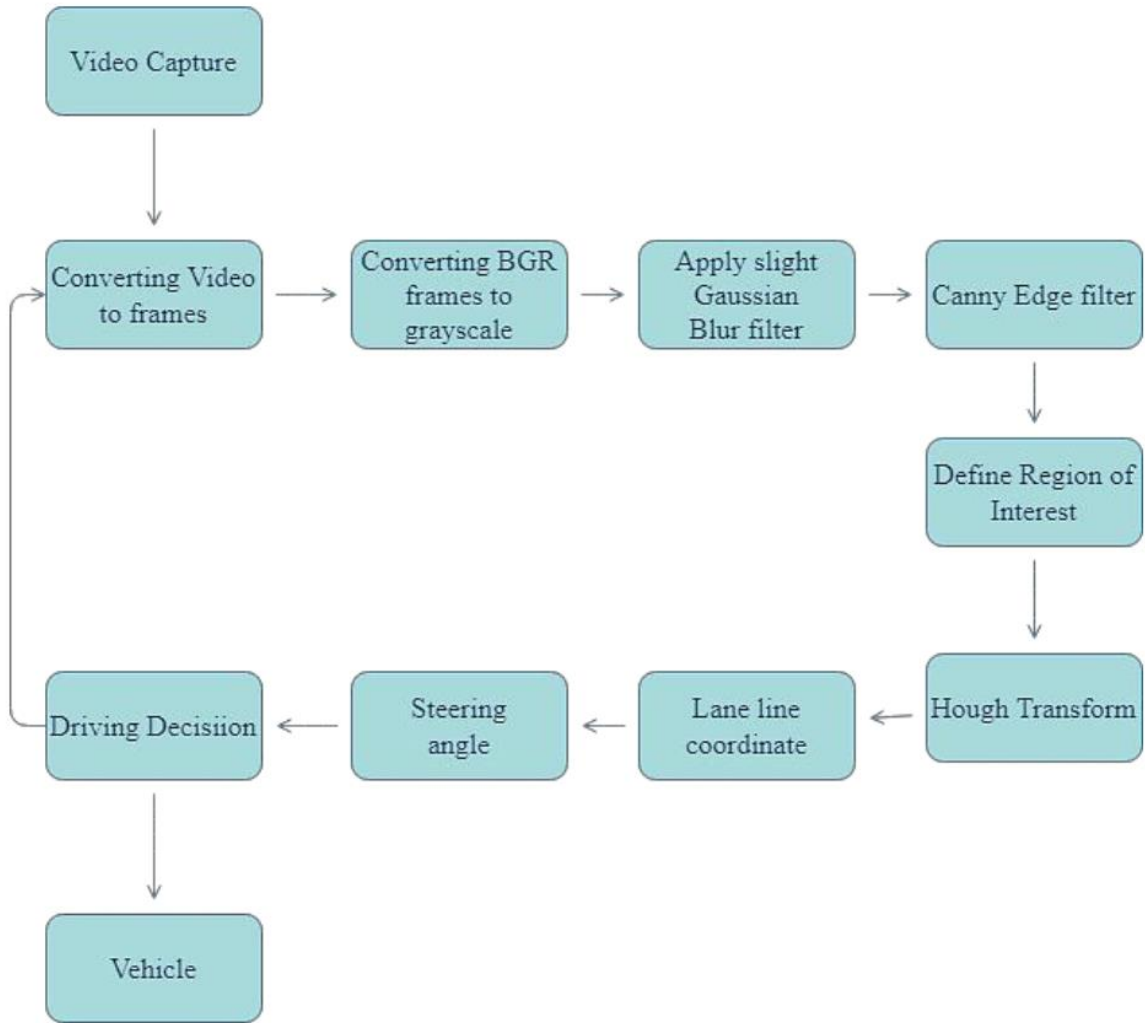


Figure 32:Algorithme de détection de ligne de voie