MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE ABDELHAMID IBN BADIS - MOSTAGANEM

Faculté des Sciences Exactes et d'Informatique

Département de Mathématiques et informatique

Filière : Informatique

PROJET DE FIN D'ETUDE

Option : **Ingénierie des Systèmes d'Information**

THEME :

Détection des voies routières par vision

Etudiante :« **Bezzaouch Meriem** »

Encadrant : « Mohammed El Amine Moumene »

Co-Encadrante : « Fatima Zohra Benidris »

Année Universitaire 2021-2022

Remerciements

C’est avec une grande gratitude et reconnaissance que je réserve ces lignes qui s’adressent à toutes les personnes qui ont contribué à l’élaboration de ce projet de fin d’études et qui m’ont aidé lors de la rédaction de ce rapport.

Je tiens en premier lieu à adresser mes remerciements à mes encadrants de mémoire Monsieur Mohammed El Amine Moumene et Madame Fatima Benidris, pour leur encadrement et leur disponibilité, ainsi que pour la richesse et la qualité de leur enseignement, mes remerciements s’adressent aussi à tous les membres du jurys pour l’honneur qu’ils m’ont fait en m’acceptant.

Je remercie également

- Tout le corps administratif de FSEI.
- Mes enseignants.
- Ma famille pour tous leurs multiples soutiens.

Résumé

Dans le cadre des véhicules autonomes sur autoroutes, l'une des premières et des plus importantes tâches consiste à localiser le véhicule sur la route. Pour cela, le véhicule doit pouvoir prendre en compte les informations de plusieurs capteurs et les fusionner avec des données issues de cartes routières.

Le problème de la perception de la route ou de la voie est un catalyseur crucial pour les systèmes avancés d'aide à la conduite. En tant que tel, il a été un domaine de recherche actif au cours des deux dernières décennies avec des progrès considérables réalisés au cours des dernières années. Le problème a été confronté à divers scénarios, avec différentes définitions de tâches, conduisant à l'utilisation de diverses modalités et approches de détection.

L’objectif de notre travail consiste à implémenter une application de détection des voies routières basée sur le fusionnement des images successives. À cet effet, nous envisageons d’utiliser l’algorithme de Canny puisqu’il a amélioré de nombreux détecteurs de contour de l'image ainsi que l’algorithme de Hough probabiliste pour que nous déduisons leur robustesse et leur capacité de détecter les voies routières. Enfin, l’étude comparative de notre approche avec et sans le prétraitement de fusionnement d’images successives a marqué que ce prétraitement améliore les performances de détection.

Mots clés

Pré-traitement des images, Extraction des caractéristiques, Ajustement de modèle, Transformée de Hough Probabiliste, Canny, Sobel, ROI, RVB, ADAS, Vision par ordinateur, RANSAC, Spline, ACO, Kalman.

Abstract

In the context of autonomous vehicles on highways, one of the first and most important tasks is to locate the vehicle on the road. To do this, the vehicle must be able to take information from several sensors into account and merge it with data from road maps.

The problem of road or lane perception is a crucial enabler for advanced driver assistance systems. As such, it has been an active domain of research over the past two decades with considerable progress being made in recent years. The problem was faced with various scenarios, with different task definitions, leading to the use of various detection modalities and approaches.

The objective of our work is to implement an application for lane road detection based on the merging of successive images. For this purpose, we plan to use Canny algorithm since it has improved many image edge detectors as well as Probabilistic Hough algorithm so that we deduce its robustness and its ability to detect lane road. Finally, the comparative study of our approach with and without the successive image fusion pre-processing showed that this pre-processing improves the detection performance.

Key words

Image pre-processing, Feature extraction, Model fitting, Probabilistic Hough transform, Canny, Sobel, ROI, RGB, ADAS, Computer vision, RANSAC, Spline, ACO, Kalman.

Liste des abréviations

ADAS	Advanced Driving Aide System
OpenCV	Open-Source Computer Vision Library
RADAR	Radio Detection and Ranging
GPS	Global Positioning Systems
LDW	Lane Departure Warning
ACC	Adaptive Cruise Control
IA	Artificial Intelligence
BMP	Windows Bitmap
PCX	PiCture eXchange
GIF	Graphic Interchange Format
JPG	Joint Photographic Group
JPEG	Joint Photographic Experts Group
DXF	Data eXchange Format
CGM	Computer Graphics Metafile
LIDAR	Light Detection and Ranging
ROI	Region of interest
IPM	Inverse perspective mapping
ACO	Ant colony optimization
FHD	Finlayson Hordley Drew
HT	Hough transform
PHT	Probabilistic Hough transform
MRDT	Multi-region dual-threshold
RANSAC	RANdom SAmple Consensus

HG Hypothesis Generation

TPU Tensor Processing Unit

GPU Graphical Processing Unit

Table des Figures

Table des matières

Introduction générale

Au cours des dernières décennies, dans le domaine des systèmes de transport, une grande attention a été accordée aux problèmes tels que l'amélioration des conditions de sécurité, optimiser l'exploitation des réseaux de transport, réduire la consommation d'énergie et préserver l'environnement. Selon les estimations de l'organisation mondiale de la santé (OMS), 1,2 million de personnes sont tuées et pas moins de 50 millions souffrent de blessures non mortelles, qui entraînent parfois des handicaps chaque année [44]. En Allemagne, plus de 200 000 accidents se produisent chaque année dans les agglomérations sont causés principalement à de fausses manœuvres de conducteurs [44].

Les efforts pour résoudre ces problèmes ont suscité l'intérêt vers un nouveau domaine de la recherche et d'application, la conduite automatique de véhicule, dans lequel de nouveaux systèmes avancés d'aide à la conduite (ADAS) sont étudiés pour l'automatisation ou semi-automatisation des différentes tâches de conduite. Ces tâches incluent le suivi de la route et maintenir une distance de sécurité entre les véhicules, la régulation de la vitesse du véhicule en fonction des conditions de circulation et les caractéristiques de la route, trouver le chemin le plus court vers une destination, le déplacement et le stationnement dans les milieux urbains et la détection d'obstacle. Les ADAS ne peuvent pas complètement prévenir les accidents, mais ils pourraient assurer la sécurité souhaitée sur les routes et mieux nous protéger de certains facteurs humains car l'erreur humaine est la cause de la plupart des accidents de la circulation.

Parmi les systèmes avancés d'aide à la conduite les plus complexes et en même temps nécessaire à la réalisation des futurs véhicules intelligents la détection des voies routières (lignes blanches sur les routes). Cette tâche consiste à détecter et reconnaitre des voies où circule le véhicule. Garder le véhicule au milieu d'une voie de circulation est cruciale pour la sécurité routière tant pour les conducteurs que pour les piétons.

Il existe une panoplie d'algorithmes de détection des voies de circulation qu'on retrouve dans la littérature. Celles-ci se basent souvent sur une phase de prétraitement, une phase d'extraction de caractéristiques et en fin d'ajustement de modèle sur les informations extraites.

L’objectif de notre projet est de réaliser un système de détection de voies routières efficace face à des conditions difficiles.

Dans ce projet, nous proposons une méthode de détection de voie en temps réel basée sur la fusion de trames successives. Cette méthode offre de bonnes performances dans la base de données Highway-Driving [62]. Tout d'abord, le prétraitement de fusion d'images successives est appliqué pour aider à connecter les marqueurs de voies en pointillés. Ensuite, l'algorithme Canny est utilisé pour extraire les caractéristiques. Par la suite, la région d'intérêt (ROI) est sélectionnée sous une forme triangulaire. Après cela, les marquages de voie possibles sont identifiés à l'aide de la transformée de Hough probabiliste. Enfin, la moyenne et l'extrapolation des lignes de voies.

Pour bien mener notre étude, nous avons réparti ce présent projet en trois chapitres :

Le premier chapitre présente les systèmes d’aide à la conduite ainsi que les méthodes de traitement d’images et de vision par ordinateur les plus utilisées afin d’en choisir les meilleures.

Le deuxième chapitre présente l’état de l’art sur les différentes approches utilisées pour la détection des voies routières avec les avantages et inconvénients de ces approches. Il se termine par l'objectif de notre projet.

Le troisième chapitre présente les étapes de notre approche proposée, les ressources utilisées, et le résultat finale de notre travail.

Nous terminons notre travail par une conclusion générale qui résume tout ce qui a été précédemment évoqué.

Chapitre 01

Aide à la conduite et traitement d’images

1.1 Introduction

Un ADAS est un système qui détecte les situations potentiellement dangereuses et avertit le conducteur afin de pouvoir éviter l'accident.

Le traitement d'image est un domaine vaste. C'est l'ensemble des techniques d'amélioration ou d'extraction des informations appliquées à l'image numérique. Ce premier chapitre a pour objectif de situer le projet dans son cadre général. Dans un premier temps, nous décrivons le domaine de l'aide à la conduite et les principales techniques de traitement d'image souvent utilisées pour la réalisation de systèmes d'assistance au conducteur.

1.2 Aide à la Conduite

La conduite n'est pas une activité bénigne. Il s'agit d'un véritable travail physique et intellectuel, qui demande de la rigueur et de la concentration. Non seulement il vous faut pouvoir accomplir plusieurs gestes à la fois (pieds sur les pédales, mains sur le volant, observation des environs), mais votre cerveau est également rudement mis à contribution. Vous devez sans cesse décoder de nombreux signaux.

Tout système facilitant la tâche du conducteur représente un outil aidant à la conduite. Ces systèmes contribuent à la sécurité du véhicule et des passagers en agissant sur la motricité du véhicule, sa capacité à accélérer, à freiner ou à maintenir une trajectoire. Ils peuvent aussi alerter en cas de perception d'un danger imminent.

1.2.1 Les types de capteurs utilisés dans les applications d'ADAS

Selon leurs objectifs, les ADAS peuvent se baser sur différents capteurs :

- **Caméras** : leur rôle est de percevoir l'environnement. Ils sont principalement utilisés en raison de leur faible coût, de leur haute résolution et de leur capacité à différencier les couleurs. L'une des utilisations des caméras dans les automobiles a été la caméra de recul, aussi appelée "caméra arrière". La figure 01 représente la caméra de recul.

Figure 1 - caméra de recul qui permet aux conducteurs de reculer en toute sécurité [8]

- **RADAR** : Le capteur Radar (Radio Detection and Ranging), qui est un appareil qui utilise des ondes radio pour la détection d'objets dans une certaine surface. Lorsque les ondes transmises interceptent un objet le long de son trajet de propagation, elles sont réfléchies par sa surface où l'antenne RADAR collecte le signal rétrodiffusé dans son champ de vision. [1]
- **LiDAR** : les systèmes LiDAR (Light Detection and Ranging) sont utilisés dans le régulateur de vitesse adaptatif, la prévention et l'atténuation des accidents et la détection d'objets. Essentiellement, le LiDAR est un type de RADAR qui utilise un ou plusieurs lasers comme source d'énergie. Et utilisé dans des applications où la détection de la forme et de la taille exacte de l'objet est nécessaire. [1]
- **GPS** : Le GPS est utilisé dans les véhicules à la fois pour le suivi et la navigation, permet d'identifier le nom actuel de l'emplacement, trouver la distance parcourue, et prédire l'heure d'arrivée. [2]

1.2.2 Quelques systèmes d'aide à la conduite

- **Lane Departure Warning (LDW)**

Avertit le conducteur quand il franchit involontairement une ligne continue ou discontinue. Les conducteurs reçoivent des avertissements sonores et visuels qui se

manifestent par une vibration dans le siège du conducteur du côté où le franchissement a lieu. [3]

- **Adaptive Cruise Control (ACC)**

Contribue à rendre la conduite sur autoroute moins fatigante. Un système ACC tente de maintenir une vitesse souhaitée définie par le conducteur tout en maintenant un écart de temps minimum entre les véhicules. Le conducteur peut neutraliser le système à tout moment en activant la pédale de frein ou d'accélérateur et peut l'activer ou le désactiver à différentes étapes du trajet. [4]

- **Lane keeping**

Aide à garder véhicule au milieu d'une voie détectée lorsqu'un changement de voie non signalé se produit en fournissant une direction ou un freinage automatique pour maintenir un véhicule dans sa voie de circulation. [5]

- **Sign assist**

Un système qui détecte les panneaux de signalisation. [6]

Figure 2 - Représentation des différents systèmes d'aide à la conduite [7]

La figure 02 représente, en haut, à gauche, système de reconnaissance des panneaux à l'aide d'une caméra et d'un GPS. En haut, à droite, affichage tête haute : informations sur la

route ainsi que des instructions et des avertissements. En bas, détection de piétons par caméra [7].

1.3 Véhicules Autonome

Une voiture autonome (parfois appelée voiture sans conducteur) est un véhicule qui utilise une combinaison de capteurs, et d'intelligence artificielle (IA) pour voyager entre les destinations sans opérateur humain.

1.4 Traitement d'images et la vision par ordinateur

La vision par ordinateur a suscité un intérêt croissant dans un large éventail de domaines de recherche ces dernières années : médicale, industriel, automobile, etc.

La vision par ordinateur désigne une technique d'intelligence artificielle permettant d'analyser des images captées par un équipement tel que la caméra. Vise à permettre aux ordinateurs de voir, d'identifier et de traiter les images de la même façon que la vision humaine faisant gagner beaucoup de temps et réduisant considérablement le taux d'erreurs humaines en utilisant des algorithmes de traitement d'images et d'apprentissage automatique.

1.4.1 Définition d'une image

D'une façon générale, l'image numérique est toute image qui a été traitée et sauvegardée sous une forme codée représentable par des nombres binaires (0 ou 1). Une image est représentée par une matrice de dimension « nombre de lignes » x « nombre de colonnes ». Chaque élément de la matrice, nommé pixel, représente l'intensité lumineuse comprise entre 0 et 255, soit 256 niveaux de gris, le niveau de gris 0 représente le noir tandis que le niveau de gris 255 représente le blanc. Autrement dit, une image est une forme discrète d'un phénomène continu obtenue après discrétisation. Cette forme est bidimensionnelle et les informations qui la présentent définissent les intensités lumineuses (couleurs ou niveaux de gris). [9] (Voir figure 3)

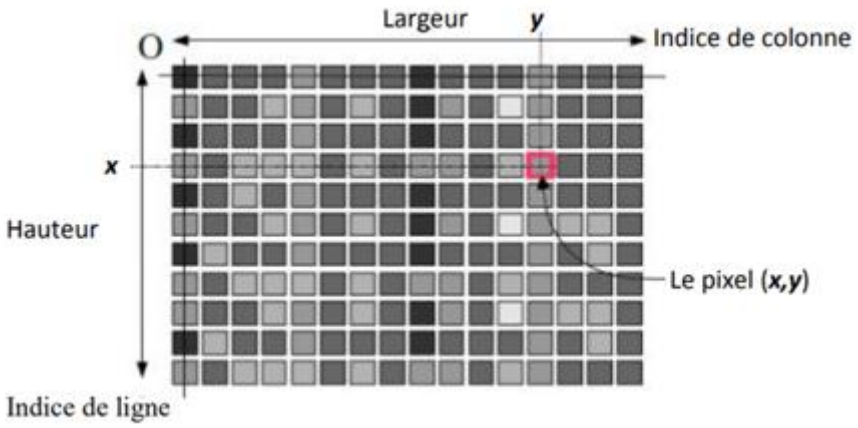

Figure 3 - Représentation d'une image numérique [10]

1.4.2 Les caractéristiques d'une image numérique

- **Pixel**

Le pixel est une unité de mesure de la définition d'une image numérique. Elle est présentée comme un petit carré de couleur. L'ensemble de ces pixels est contenu dans un tableau à deux dimensions. [19]

- **Résolution**

C'est le nombre de points contenu dans une longueur donnée en ppp (point par pouces) ou en dpi (dots per inch). Plus la résolution est grande, plus votre image est précise dans les détails [11]. Notons que : 1 pouce (ou inch) = 2,54 cm.

- **Dimension**

La définition est le nombre de points (ou pixels) que comporte une image numérique en largeur et en hauteur (le nombre de colonnes et nombre de lignes). On parle aussi de Taille en pixels. Exemple : une image dont la définition est 1600x1200 correspond à une image de 1600 pixels en largeur et 1200 pixels en hauteur.

- **Profondeur**

La profondeur (poids de chaque pixel) de l'image est le nombre de bits par pixel, cette valeur reflète le nombre de couleurs ou de niveaux de gris d'une image. [12] Par exemples :

32 bits/pixel = 1,07 milliards de couleurs

24 bits/pixel = 16,7 millions de couleurs

16 bits/pixel = 65 536 couleurs

8 bits/pixel = 256 couleurs

- **Poids**

Représente la quantité de mémoire nécessaire pour stocker une image non compressée. Le poids de l'image est alors égal à sa dimension multipliée par sa profondeur.

- **Bruit**

Un bruit (parasite) dans une image est considéré comme un phénomène de brusque variation de l'intensité d'un pixel par rapport à ses voisins, il provient de l'éclairage des dispositifs optiques et électroniques du capteur. [10]

- **Contraste**

L'opposition marquée entre les régions sombres et les régions claires d'une image.

- **Luminance**

C'est le degré de luminosité des points de l'image. Elle est définie aussi comme étant le quotient de l'intensité lumineuse d'une surface par l'aire apparente de cette surface, pour un observateur lointain, le mot luminance est substitué au mot brillance, qui correspond à l'éclat d'un objet. [10] Une bonne luminance se caractérise par : L'absence de parasites et éviter les images de contraste qui tend vers le blanc ou le noir.

- **Histogramme**

Représente la répartition des pixels en fonction de leur niveau de gris. Il fournit diverses informations comme les statistiques d'ordre, et peut permettre d'isoler les objets. [13]

- **Texture**

C'est la répartition statistique ou géométrique des intensités dans l'image. [13]

1.4.3 Différents formats d'image

- **L'image Matricielle (bitmap)**

Il s'agit d'images pixellisées, c'est-à-dire un ensemble de points (pixels) contenus dans un tableau, chacun de ces points possédant une ou plusieurs valeurs décrivant sa couleur.[12]

Les formats standards des images matricielles : BMP (Windows Bitmap), PCX (PiCture eXchange), GIF (Graphic Interchange Format), JPG ou JPEG (Joint Photographique Experts Group). [11]

- **L'image vectorielle**

Les données sont représentées par des formes géométriques simples qui sont décrites d'un point de vue mathématique. Par exemple, un cercle est décrit par une information du type (cercle, position du centre, rayon). Ces images sont essentiellement utilisées pour réaliser des schémas ou des plans. [12]

Des formats standards des images vectorielle : DXF (Data eXchange Format), CGM (Computer Graphics Metafile), etc. [11]

1.4.4 Les types des images

- **Image noire et blanc**

Image qui n'a que deux valeurs possibles pour chaque pixel. Chaque pixel est soit noir, soit blanc. Il faut alors un seul bit pour coder un pixel (0 pour noir, 1 pour blanc).

- **Image en niveau de gris**

Les pixels de l'image sont représentés sur 8 bits (1octet). Nous avons alors 256 possibilités (nous disons 256 nuances de gris). La valeur 0 correspond au noir et 255 correspond au blanc. Les valeurs intermédiaires sont plus ou moins gris foncé.

- **Image en couleur**

Chaque pixel reçoit directement les valeurs des trois canaux RVB (rouge-vert-bleu, RGB), chaque composante RVB exige un octet (8bits) pour pouvoir afficher 256 intensités différentes du canal, chaque pixel sera donc représenté par 24 bits. Ceci nous permet d'obtenir en fin du compte 16,777,216 couleurs différentes. [9]

1.4.5 Les principales techniques de traitement des images

a) Acquisition

Avant de traiter l'image il faut réaliser d'abord ces images, c'est l'étape de l'acquisition qui consiste à transformer les vues réelles en des images numérique. Le passage de cet objet externe (l'image d'origine) à sa représentation interne (dans l'unité de traitement) se fait grâce à une procédure de numérisation (échantillonnage, quantification).

- **Échantillonnage :** Le pixel peut prendre une valeur unique p (x, y).
- **Quantification :** Elle désigne la limitation du nombre de valeurs différentes que peut prendre un pixel p (x, y), ne dépasse pas 255.

b) Filtrage

En pratique, il s'agit de créer une nouvelle image en se servant des valeurs des pixels de l'image d'origine. D'une façon générale, le filtrage est obtenu par convolution de l'image avec un noyau défini. Ce noyau peut être interprété comme une petite image ou vignette contenant un gabarit de transformation et que l'on applique sur chacun des pixels de l'image à filtrer pour créer une nouvelle image.

Il y a plusieurs types de filtrage parmi eux :

- **Filtre median**

Le filtre médian est très efficace pour supprimer le bruit impulsif. Le principe du filtre médian est de remplacer le niveau de gris de chaque pixel par la médiane des niveaux de gris dans un voisinage des pixels, au lieu d'utiliser l'opération moyenne (voir la figure 4). Généralement il utilise un noyau 3x3. [14]

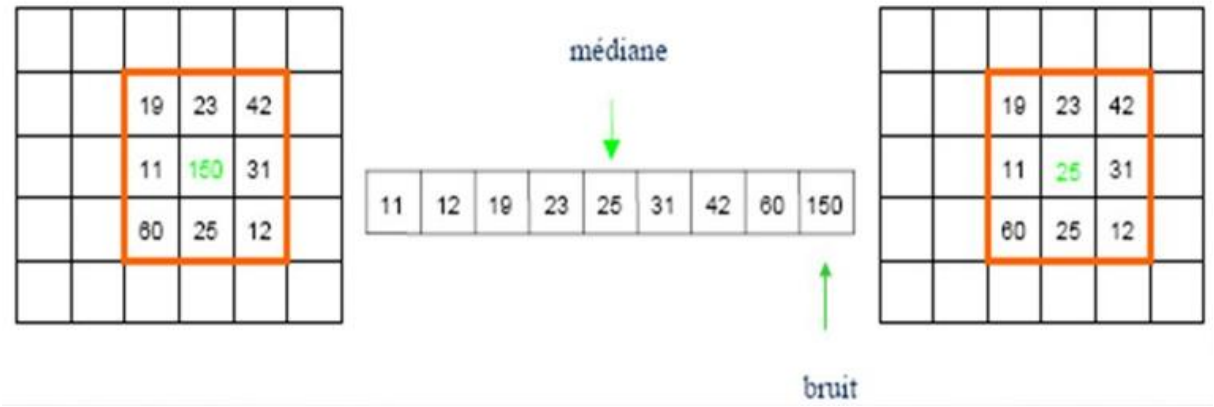

Figure 4 - fonctionnement du filtre median [15]

c) Segmentation

Elle permet d'isoler dans l'image les objets sur lesquels doit porter l'analyse et de séparer les régions d'intérêt du fond. Il existe plusieurs techniques, la plus simple étant le seuillage. Plus la segmentation est meilleure, plus l'étape de reconnaissance d'objets est réussie.

d) Seuillage

Est le traitement qui permet de sélectionner les informations significatives dans les images. L'image d'entrée est une image en niveau de gris et l'image résultant de ce traitement est en noir et blanc.

Ce traitement nécessite le réglage d'un paramètre : le seuil S. Si la valeur du pixel de l'image dépasse le seuil fixé, la valeur résultante du pixel est 1, sinon la valeur résultante du pixel est 0.

Il existe trois méthodes pour calculer le seuil :

- **Seuillage fixe :** est le plus simple. Il fonctionne comme suit :

1. Prendre un seuil fixe en paramètre de la fonction.
2. Parcourir tous les pixels de l'image, si le pixel est supérieur à ce seuil il prend la valeur 1 sinon il prend la valeur 0.

- **Seuillage global :** fait appel à la fonction de seuillage fixe mais en fixant le seuil.

Le seuil est la moyenne globale des pixels de l'image. Additionner tous les pixels pour diviser le tout par le nombre de pixels de l'image originale. La moyenne est trop faible pour obtenir des contours corrects.

- **Seuillage local :** calcul aussi une moyenne mais une moyenne locale au pixel courant.

Passer en paramètre un voisinage qui sert à déterminer quels pixels doivent être utilisés pour calculer cette moyenne. Avec un voisinage de 1 on calcule la moyenne avec les 8 points qui entourent le pixel courant. Ce seuillage n'est pas très efficace parce qu'on fait une moyenne locale et non plus globale.

e) Détection des contours

La détection des contours est un outil de base utilisé dans le traitement d'images, essentiellement pour la détection et l'extraction de caractéristiques, qui vise à identifier les points d'une image numérique où la luminosité de l'image change fortement et à trouver des discontinuités. Le but de la détection des contours est de réduire considérablement la quantité de données dans une image et de préserver les propriétés structurelles pour un traitement ultérieur de l'image. Dans une image en niveaux de gris, le contour est une caractéristique locale qui, dans un voisinage, sépare des régions dans chacune desquelles le niveau de gris est plus ou moins uniforme avec des valeurs différentes des deux côtés du contour. Pour une image bruitée, il est difficile de détecter les contours car les contours et le bruit contiennent des contenus à haute fréquence qui entraînent un résultat flou et déformé. La figure 05 représente les différents types de détecteur de contour. [17]

Il existe plusieurs méthodes de détection de contour, l'approche la plus utilisée est :

Approximations du Gradient qui comprend deux méthodes de detection de contour sont les opérateurs classiques et l'algorithme de Canny.

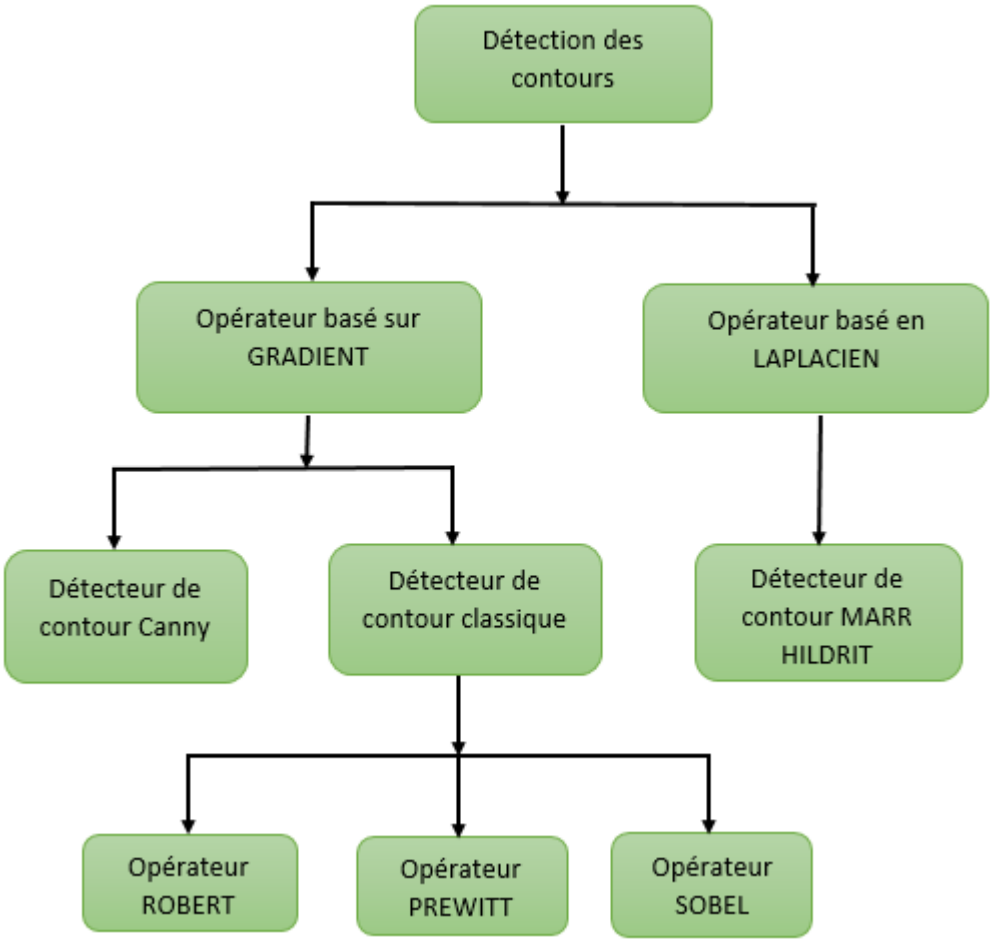

Figure 5 - Types de détecteur de contour [18]

1) Opérateurs classiques

Robert, Sobel, Prewitt sont classés parmi les opérateurs classiques faciles à utiliser mais très sensibles au bruit [19]. La figure 06 représente l'organigramme général pour les opérateurs classiques.

- **Opérateur Sobel :**

L'opérateur de Sobel est un opérateur de différenciation discret utilisé pour calculer une approximation du gradient de la fonction d'intensité de l'image pour la détection des contours. Il convolue l'image d'entrée avec le noyau et calcule l'amplitude et la direction du gradient [19]. Il utilise les deux filtres 3x3 suivants :

$$D_i = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \quad D_j = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

Et qui suit seront les différentes étapes de l'opérateur Sobel :

- Di et Dj deux images qui en chaque point contiennent des approximations respectivement de la dérivée horizontale et verticale de chaque point et si F (i, j) est l'image d'entrée :

$$G_i = D_i * F(i,j) \; Et \; G_j = D_j * F(i,j) \qquad (1)$$

- La force des contours ou l'amplitude du gradient pour chaque pixel calculé par la formule (2) :

$$\sqrt{Gi^2 + Gj^2} \qquad (2)$$

- Direction du gradient pour chaque pixel calculé par la formule (3) :

$$\theta = arctan\left(\frac{G_j}{G_i}\right) \qquad (3)$$

Et Gi et Gj sont les gradients dans les directions i et j respectivement.

- Et la dernière étape comparer la force du contour 'G' avec un seuil choisi 'S', et si G>S considérer ce pixel comme un contour.

- **Opérateur robert :**

La fonction du détecteur de contour robert est la même que celle du détecteur Sobel mais a des filtres différents [19] :

$$D_i = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad D_j = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

- **Opérateur Prewitt :**

La fonction du détecteur de contour Prewitt est la même que celle du détecteur Sobel mais a des filtres différents [19] :

$$D_i = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix} \quad D_j = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix}$$

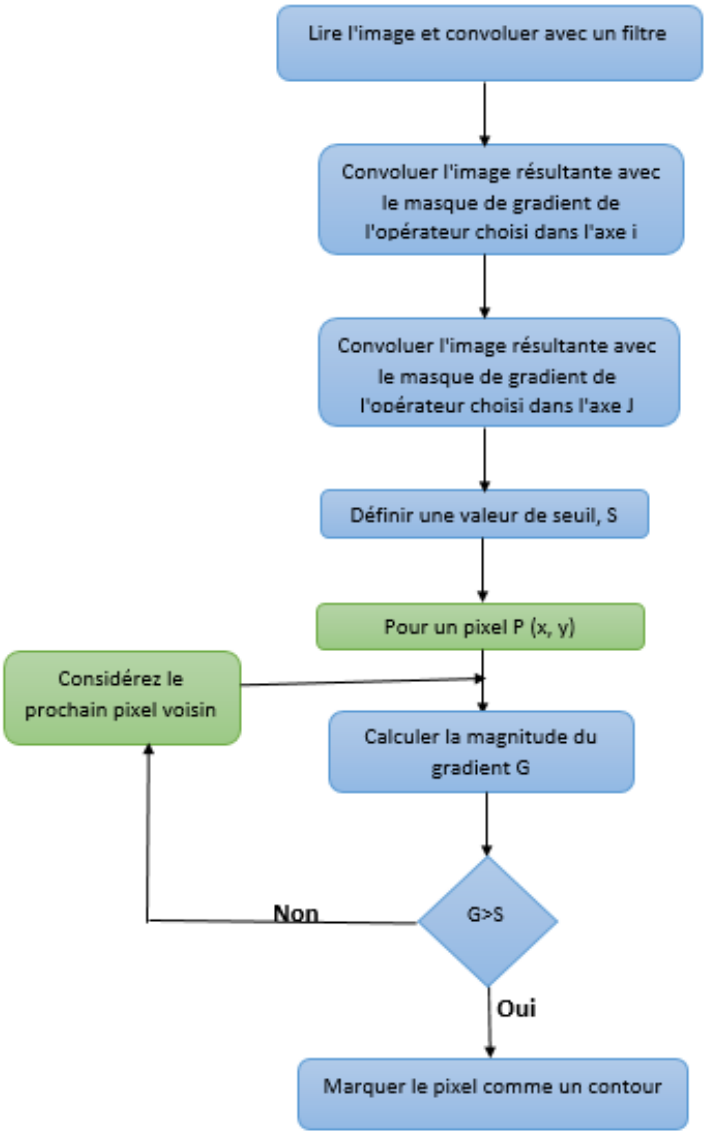

Figure 6 - Organigramme de l'algorithme général pour les opérateurs classiques [18]

2) Algorithme de Canny

L'algorithme de détection de contour Canny (Détecteur de contour astucieux) est également connu comme le détecteur de contour optimal. Les intentions de Canny étaient d'améliorer les nombreux détecteurs de contour de l'image.

Le détecteur de contour astucieux lisse d'abord l'image pour éliminer le bruit par le filtre de gaussien. Il trouve ensuite le gradient de l'image à l'aide d'opérateurs de détection des contours (Sobel généralement) pour mettre en évidence les régions avec des dérivées spatiales élevées. L'algorithme suit ensuite ces régions et supprime tout pixel qui n'est pas au maximum en utilisant une suppression non maximale. Le réseau de dégradés est encore réduit par hystérésis pour supprimer les rayures et l'amincissement des contours [20].

1.5 Problématique

La localisation d'un véhicule dans les voies de circulation est une thématique étudiée depuis plusieurs décennies par la communauté scientifique afin de répondre à diverses problématiques telles que l'entretien de l'infrastructure routière ou le contrôle latéral d'un véhicule autonome. A ce jour, les systèmes de détection des voies routières souffrent de difficultés aux mauvaises conditions routières, la non présence de marquage au sol ou leur occlusion.

Afin de traiter les problèmes mentionnés ci-dessus résultant de changements dans les limites des voies et déceler la meilleure approche dans le but d'implémenter une application robuste face aux situations adverses nous discutons dans le deuxième chapitre les différents travaux de recherche sur la détection de voies routières.

1.6 Conclusion

Nous avons débuté notre chapitre avec une présentation de l'aide à la conduite et les systèmes ADAS qui ont un potentiel énorme pour augmenter la sécurité, de nos véhicules et les techniques de traitement d'image.

Le chapitre suivant sera consacré sur quelques travaux existants qui utilisent différentes techniques pour la détection des voies routières.

Chapitre 02

Etat de l’art sur la détection des voies routières

2.1 Introduction

Ce chapitre présente un aperçu de la littérature existante. Pour les analyses de la détection des voies routières, une littérature importante provenant de sources multiples est référencée. Les auteurs ont adopté les approches les plus courantes pour la détection des voies de circulation impliquant la transformation de Hough, la détection des contours Canny avec Hough, RANSAC, ACO, etc.

2.2 Travaux relatifs

La détection des voies routières est un domaine de recherche qui a attiré l'attention de nombreux chercheurs durant les deux dernières décennies. Différentes méthodes de vision par ordinateur sont appliquées pour la détection des marquages routiers et on distingue dans ces méthodes trois phases incontournables : le prétraitement, l'extraction des caractéristiques et l'ajustement de modèle sur les points extraits.

2.2.1 Pré-traitement des images

Cette phase représente la partie qui prépare les images pour la phase d'extraction des caractéristiques en appliquant différentes techniques de traitement d'images.

1) **Algorithmes basés sur la suppression d'éclairage et d'ombre**

- Algorithme de seuillage et utilisation des filtres

De nombreux chercheurs, utilisent le seuillage de l'image avec un seuil prédéfini ou un seuil adaptatif. D'autres, appliquent des filtres orientables comme le filtre médian et filtre gaussien pour supprimer le bruit (voir chapitre 1), et Finlayson-Hordley-Drew (FHD) pour supprimer les ombres fortes de l'image. [48]

- Transformations de l'espace colorimétrique

Dans [25,26,27] effectuent une variété de transformations de l'espace colorimétrique RVB en HSI, LAB, YCbCr et autres. Le résultat de cette transformation est d'obtenir des images invariantes à l'éclairage où les zones éclairées et ombrées de la même surface obtiennent une intensité similaire. Soleil et al [48] assurent que la détection des voies donne de bons résultats en utilisant un modèle HSI par rapport au RVB et les images monochromes sont préférées pour les routes structurées, en raison d'une meilleure résolution et d'une charge de données réduite.

Figure 7 - Exemple de prétraitement d'images [56]

Figure 7 représente le résultat de prétraitement d'image. L'image est prétraitée en la convertissant en une image en niveaux de gris, puis assombrie afin d'avoir plus d'espace entre les parties les plus sombres et les plus claires de la scène.

- Méthode basée sur histogramme

Bottazzi et al, proposent une méthode de détection de voie invariante par illumination basée sur un histogramme. L'histogramme de l'image entière et le cadre de la route sont calculés, la différence entre les deux est utilisée pour découvrir les changements d'éclairage. [57]

- Déduire l'emplacement du soleil

Dans [24], ont résolu le problème d'éclairage causée par la lumière directe du soleil dans le champ de vision de la caméra. La date, l'heure et les coordonnées géographiques sont utilisées pour le calcul d'une éphéméride solaire permet de déduire l'emplacement du soleil sur le plan de l'image et de rejeter les lignes droites lumineuses pointant dans cette direction.

2) Sélection de la région d'intérêt ROI

L'élagage des parties de l'image qui sont suspectées d'être non pertinentes consiste à définir des régions d'intérêt (ROI) qui seront traitées lors de la phase d'extraction des caractéristiques. Certains chercheurs ont préféré diviser l'image par un pourcentage prédéfini à partir du bas ou du haut et cette partie de l'image est considérée comme région d'intérêt, certains ont pris la partie entre le bas de l'image et le point de fuite qui représente le croisement des voies dans l'image de la scène routière, et d'autres divisent l'image horizontalement ou verticalement en plusieurs parties [32].

3) Algorithme basée sur la méthode IPM (Inverse Perspective Mapping)

Dans [55] appliquent Inverse Perspective Mapping (IPM) permet de reproduire une image représentant la route vue d'en haut (vue d'oiseau). IPM permet de supprimer l'effet de

perspective de l'image ce qui permet de visualiser les bords de la route comme étant des lignes droites parallèles. La figure 8 représente le résultat de cette transformation.

- **Avantages de IPM** : Un certain nombre d'avantages sont consacrés par cette technique. La transformée de l'image et la recherche des lignes verticales imposent en soi la contrainte que les marquages de la route soient parallèles (contrainte de point de fuite). La robustesse de la technique en présence d'ombres est réalisée par la recherche des régions foncée-claires qui représente des bandes peintes sur la chaussée. La transformée en perspective inverse (IPM) permet l'utilisation d'algorithme rapide de traitement d'images ce qui permet un traitement en temps réel. [59]

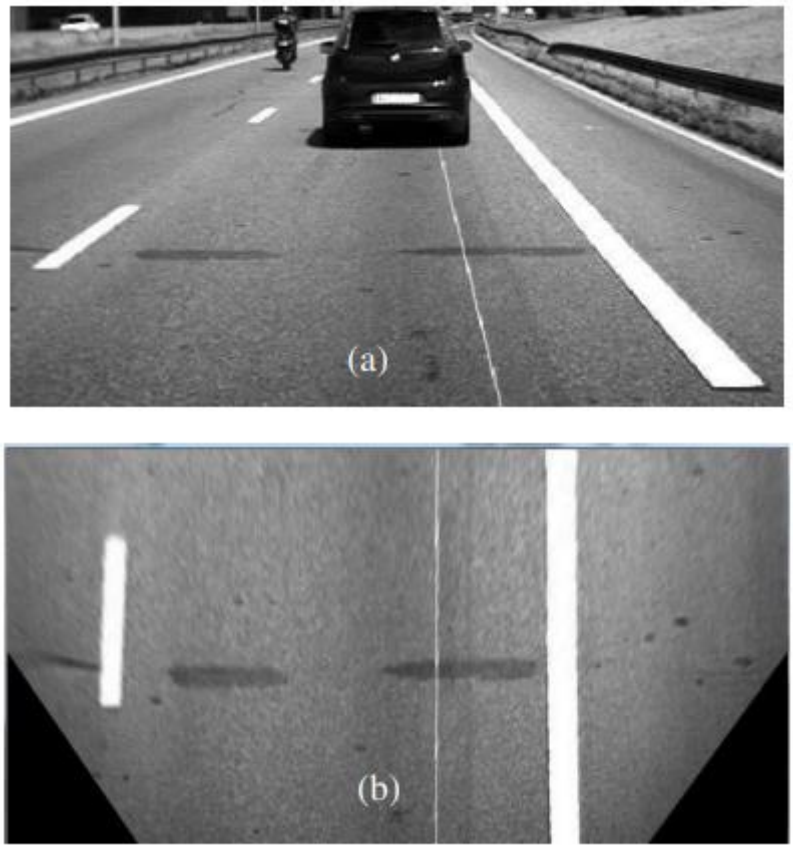

Figure 8 - La transformée inversée par la méthode IPM, (a) Image vue de scène, (b) Image vue d'oiseau. [58]

4) Réduction de la taille de l'image

Dans [25], la taille de l'image capturée par la caméra a été réduite par sous-échantillonnage en raison du coût élevé du traitement d'image à haute résolution, mais dans des situations complexes le changement de la taille des images entraînait fréquemment la perte d'informations utiles.

2.2.2 Extraction des caractéristiques

Cette phase représente l'extraction des caractéristiques qui contient des informations requises pour la phase d'ajustement du modèle de voie qui suivra.

1) Détection des contours

Dans [48], utilisent le filtre Canny, filtres classiques de Laplacian, Sobel, Robert, et Prewitt. (Voir chapitre 1)

Figure 9 - Résultat du détecteur de contour Canny [56]

Figure 9, représente en haut l'image original, et en bas résultat de l'algorithme Canny qui permet la détection des contours des lignes de voie. C'est là que commence la reconnaissance réelle des voies.

- **Avantages de détecteur de contour Canny**

Moins sensible au bruit car il utilise un filtre gaussien qui le supprime dans une large mesure, ne pas manquer le vrai contour, de nature adaptative ce n'est pas comme les opérateurs classiques qui utilisent des noyaux fixes [20].

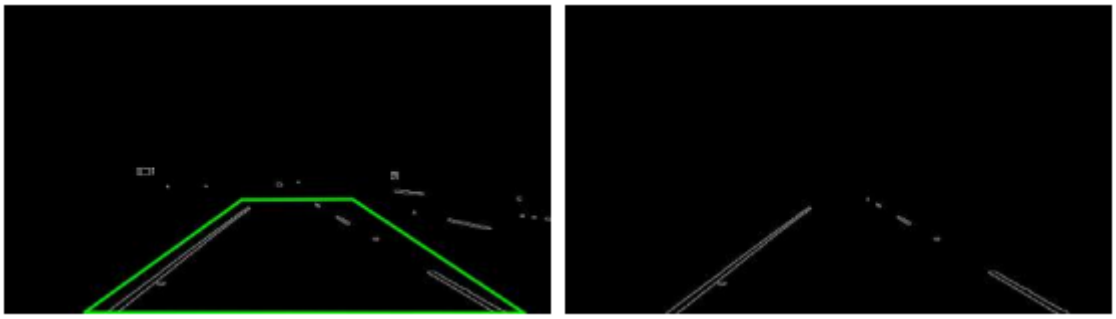

Figure 10 - Sélection de la région d'intérêt ROI [56]

Figure 10 représente dans ce cas la détection de ROI après la détection de contours. En termes simples, nous allons sélectionner un polygone décidé à l'avance. Ce polygone définira la région de l'image dont nous avons besoin. Par conséquent, les autres pixels de l'image sont ignorés.

2) Détection des contours avec optimisation des colonies de fourmis

Dans [60], Daigavane et al ont développé une méthode de détection des voies basée sur l'optimisation des colonies de fourmis (ACO) qui est inspirée du comportement des fourmis, le but de ACO est d'extraire les informations de contour supplémentaires qui sont ignorées lors de la détection de contour de Canny. Les fourmis sont placées initialement aux extrémités des segments de contour produits par l'algorithme de détection de contour Canny, mais les valeurs d'intensité d'origine de l'image sont utilisées pour guider les fourmis pour se déplacer sur l'image entraînée, pour établir une matrice de phéromones qui représente les informations de contour à chaque emplacement de pixel de l'image. Pour chaque fourmi au cours de l'itération, nous déplaçons continuellement la fourmi en utilisant une approche probabiliste jusqu'à ce que la fourmi atteigne un autre segment de ligne. En construisant les informations sur les phéromones sur l'espace des solutions, ACO vise à trouver la solution optimale du problème cible grâce au mouvement d'un certain nombre de fourmis (Voir figure 11).

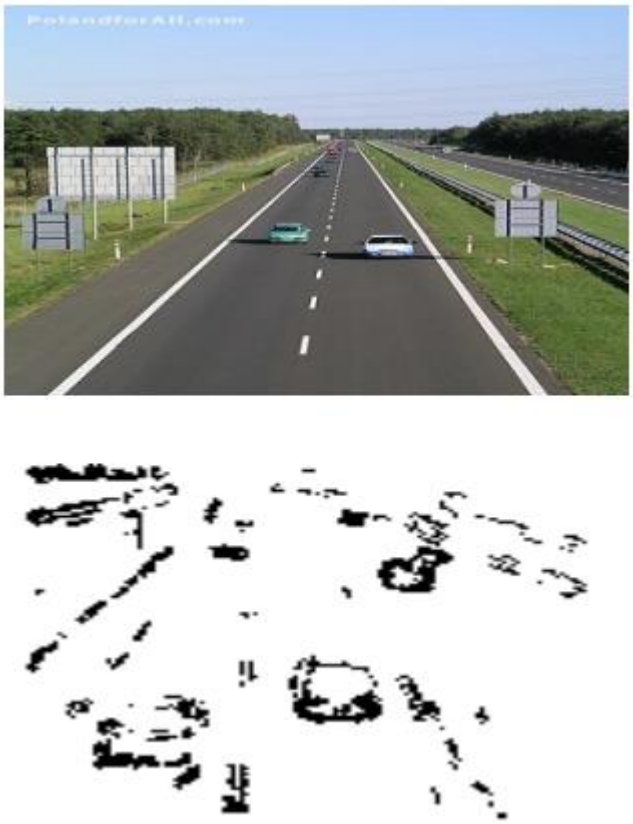

Figure 11 - Détection des contours avec optimisation des colonies de fourmis [60]

La figure 11 représente en haut l'image original, et en bas la détection des contours avec optimisation des colonies de fourmis.

3) Traiter les contours superflus

Parfois les techniques basées sur les contours peuvent échouer, dans ces cas des chercheurs utilisent des techniques avérées efficaces pour traiter les contours superflus.

- Algorithme multi-régions à double seuil (MRDT)

Cette méthode combine le seuil de niveau de gris dans l'image d'origine et l'opérateur Canny pour supprimer les contours superflus. Cette image est d'abord divisée en un petit nombre de sections horizontales. Après cela, l'algorithme traite chaque section avec un seuil différent. [53]

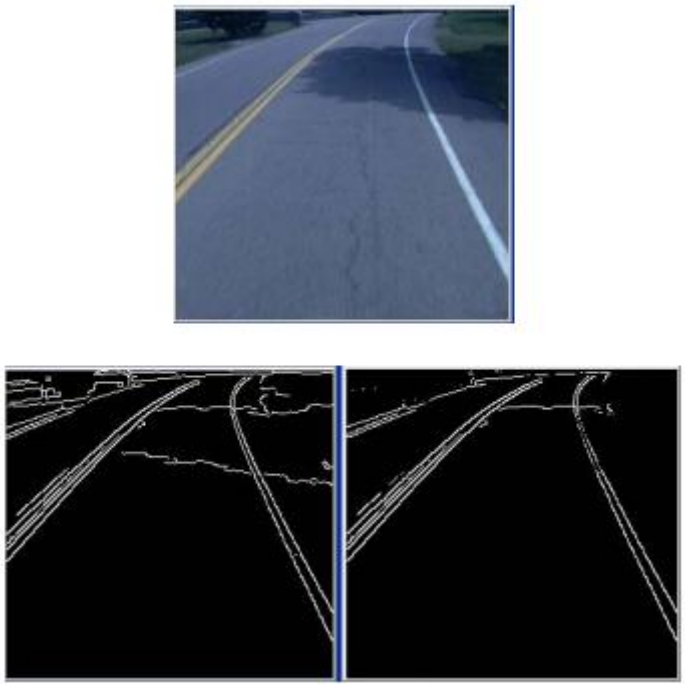

Figure 12 - Détection des contours en utilisant MRDT [53]

Figure 12, représente en haut l'image original, en bas à gauche résultat de detection de contours avec Canny, et en bas à droite résultat après l'application de l'algorithme MRDT.

2.2.3 Ajustement de modèle

Cette phase représente la détection des voies routières basée sur la connaissance préalable de la route et des caractéristiques extraites.

Les algorithmes proposés dans la littérature pour la détection des voies routières :

1) Transformée de Hough (HT)

La transformée de Hough est une technique générale pour identifier la location et l'orientation de certaines forme géométriques dans une image numérique parmi eux les lignes, HT est la technique la plus efficace pour la détection et l'estimation des voies avec des contraintes de mémoire réduites. Les performances des algorithmes HT dépendent fortement de la quantité de données.

Dans [49], les auteurs ont proposé d'autres HT particulières comme la Transformée de Hough randomisée (RHT) et la transformée probabiliste de Hough (PHT) qui résout le même problème avec un faible coût de calcul.

- **Avantage de HT :** La transformée de Hough permet de localiser n'importe quelle ligne qui peut être décrite par une équation analytique. Elle est robuste (bruit, occlusions).
- **Inconvénient de HT :** Le temps de calcul important.

Figure 13 - Transformation de Hough pour la détection des lignes de voie [56]

Figure 13, représente le modèle de détection des voies routières à l'aide de transformée de Hough.

2) Détection de plusieurs voies de circulation (Multi-Lane detection)

Le principe de cet algorithme est basé sur deux principales phases : la détection des lignes de la route (trois lignes au minimum), l'estimation de trois lignes (génération d'hypothèses HG) qui sont détectées lors de la phase précédente. [24, 28]

- Détection des lignes

À l'aide de l'algorithme de RANSAC (RANdom SAmple Consensus) qui est une technique basée sur le principe de génération et de vérification d'hypothèses. Étant donné un modèle avec des points de contours sélectionnés au hasard et en supposant un modèle de ligne, l'algorithme RANSAC détermine les lignes les plus probables et ignore les valeurs aberrantes.

- Estimation des lignes adjacents par hypothèse (HG)

Nous supposons que toutes les lignes de la route sont parallèles et ont la même largeur, après, nous transformons l'image en vue d'oiseaux, et enfin on déduit les lignes adjacentes par la méthode de cross-ratio. [24, 28]

- **Avantage de RANSAC** : Les modèles polynomiaux ont la capacité d'estimer les paramètres de la route et insensible aux valeurs aberrantes et il a la capacité de détecter plusieurs voies de circulation (au moins trois lignes). [50]
- **Inconvénient de RANSAC** : Ne peut pas gérer un changement brusque de courbure. Les hypothèses géométriques ne sont pas toujours correctes (par exemple, prendre 3 à 3,5 m comme voie de largeur). [50]

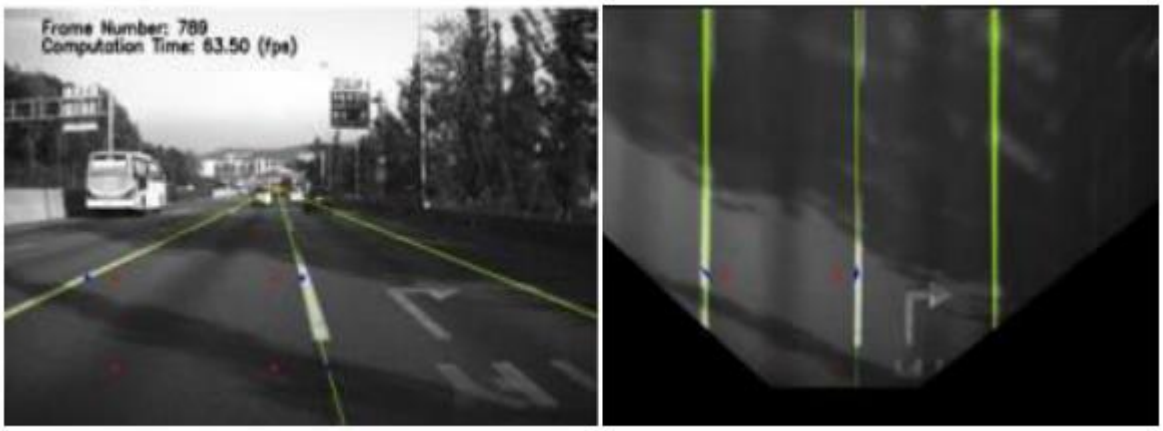

Figure 14 - Détection des voies routières en utilisant RANSAC. [58]

Figure 14, représente la détection des voies routières en utilisant RANSAC. À gauche image normale, à droite image en vue d'oiseaux.

3) K-Means Clustering

Dans [52] ont combiné l'Inverse Perspective Mapping (IPM) et le K-Means Clustering afin de regrouper les données similaires en groupes pour trouver les voies correctes, et un HT pour détecter la ligne dans l'image.

- **L'avantage d'utiliser K-Means** : est qu'il nous fournit une sortie remarquablement précise de manière cohérente. L'erreur de l'algorithme dépend en fait de l'erreur des données disponibles (erreur générée à partir des transformations de Hough) qui sont assez précises par elles-mêmes. [52]

4) Splines

Les splines sont des fonctions polynomiales, et elles sont largement utilisées pour représenter des courbes, et adopte le modèle 2D parabolique. Les étapes de l'algorithme sont :

1. L'extraction des contours en utilisant le filtre de Canny.
2. La détection des lignes droites par la transformée de Hough.
3. Détection du point de fuite.
4. L'estimation du centre de la route.
5. L'initialisation des points de contrôle qui sont les paramètres qui donnent la forme à la courbe. (Voir figure 15)

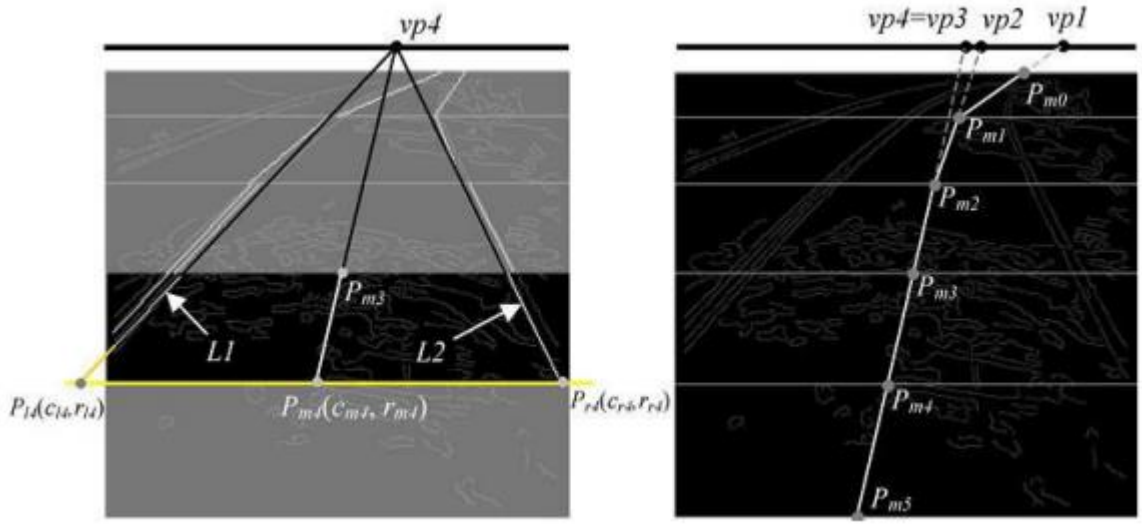

Figure 15 - L'estimation des points de contrôle. [58]

Figure 16 - résultat final de l'algorithme Spline [58]

Figure 16, représente résultat de détection des voies routières en utilisant les splines.

- **Avantage de Spline** : C'est le petit changement dans la courbe, ce qui facilite l'utilisation de points de contrôle suivis de l'image précédente pour l'initialisation du modèle dans l'image actuelle. Capable de gérer une large gamme de routes courbes à l'aide de points de contrôle s'ils sont choisis avec précision. [45]
- **Inconvénient de Spline :** L'inconvénient de ce modèle apparaît dans le choix des points de contrôle. La position de ces points de contrôle affectera la courbe générale de la voie. Un mauvais choix de ces points de contrôle conduit à une forme de route irréaliste. [45]

5) Filtre Kalman

Dans [51], ont présenté un algorithme pour détecter les voies de circulation basée sur trois étapes. Dans la première, la luminosité est utilisée dans l'image d'entrée pour connaître les conditions environnementales. Dans la deuxième étape, la HT est appliquée afin d'identifier les

lignes de route droite et gauche. Dans la troisième étape, un filtre de Kalman est appliqué pour estimer la position du véhicule et suivre la voie.

Le filtre Kalman utilise deux étapes majeures, une étape de prédiction et une étape de correction. L'étape de prédiction permet de prédire le dernier état estimé. L'étape de correction utilise une mesure pour corriger l'état prédit. Nous pouvons dire que le filtre de Kalman ne croit pas ce qui est observé, mais il essaie d'observer ce qu'il croit et qu'il corrige sa conviction avec une certaine quantité de ce qui est observé.

- **Avantage de Kalman** : Peut gérer les situations où il y a un changement brusque (par exemple, à la jonction de routes droites et courbes) et robuste au bruit. [54]

2.3 Objectif de notre travail

L'objectif de notre travail est de concevoir et d'implémenter une application qui permet de détecter les voies routières pour un environnement routières sécurisé. Notre approche divisée en trois grandes étapes, la première est le prétraitement des images qui est le fusionnement des images successives. La deuxième étape est la détection des contours à l'aide de l'algorithme de Canny dans le but d'extraire les caractéristiques des marquages routiers et préserver les données nécessaires à analyser. La troisième étape est la détection des voies routières à l'aide de transformée de Hough probabiliste et une étude comparative entre notre méthode proposée sans et avec le prétraitement de fusionnement des images successives sera effectuée. En utilisant le langage de programmation Python et la bibliothèque OpenCV.

2.4 Conclusion

Dans ce chapitre, nous avons fait une collecte d'informations qui nous a permis de préparer une étude théorique au cours de laquelle nous avons étudié les méthodes des principaux chercheurs qui proposent et implémentent des algorithmes efficaces et robustes pour détecter et suivre les voies routières.

Dans le chapitre suivant, nous allons nous concentrer sur les différentes parties de modélisation et d'implémentation pour le développement d'une application de détection des voies routière basée sur le fusionnement d'images successives.

Chapitre 03

Conception et Implémentation

3.1 Introduction

Ce chapitre porte sur la construction d'une application qui permet de reconnaitre des voies de la route en utilisant les différentes technologies et d'algorithmes de traitement d'image. Nous allons présenter la partie mise en œuvre et nous décrivons l'algorithme adopté, son architecture générale, et les étapes de développement employées pour réaliser notre approche. Nous définissons les outils qui ont permis d'accomplir notre travail, par la suite, nous présentons ces résultats.

3.2 Méthodologie de l'algorithme

Un tel système intelligent de détection des voies routières basé sur la fusion de trames successives, exige un algorithme qui répond à des conditions de la bonne performance, dans ce but nous allons discuter notre approche agréée, au premier, nous commençons par appliquer un prétraitement qui fusionne les images successives en une séquence vidéo. Après cela, les caractéristiques sont extraites en fonction de Canny. Ensuite, la région d'intérêt qui contient uniquement la partie des marqueurs de voie est sélectionnée. Après, un ajustement de modèle basé sur la transformée de Hough probabiliste est utilisé pour détecter les voies. Enfin, la moyenne et l'extrapolation des lignes de voies.

3.3 Architecture générale de l'algorithme

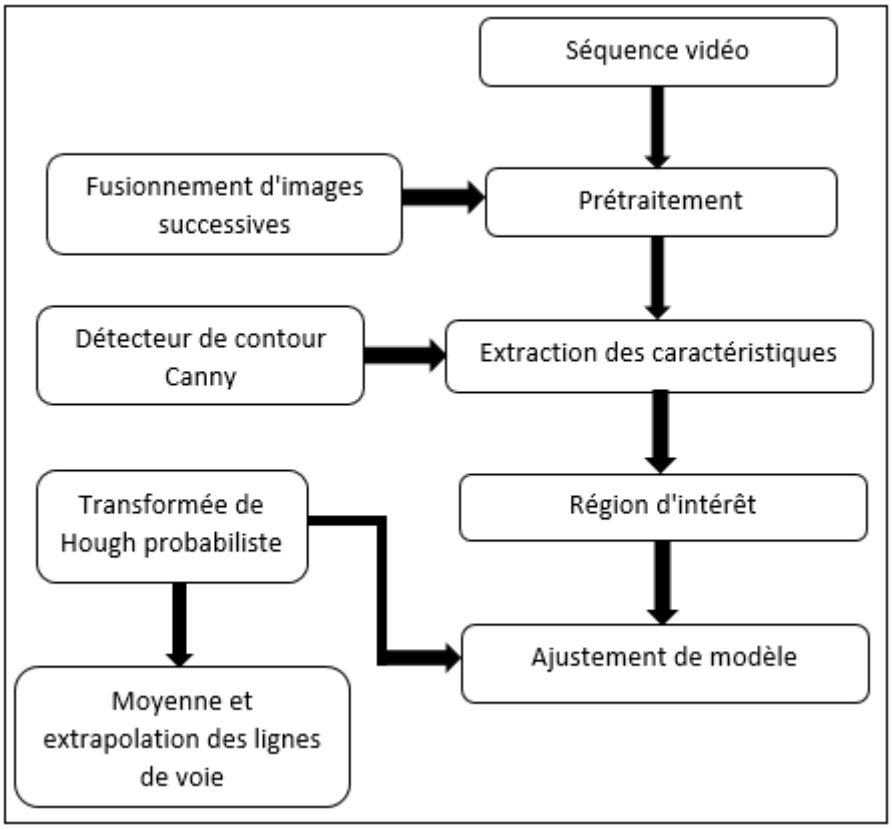

Figure 17 - L'architecture générale de l'algorithme.

3.3.1 Base de données

La base de données Highway Driving [62] est une collection de vidéos et leurs annotations pour la segmentation vidéo sémantique. La base de données a été collectée dans un scénario de conduite, ce qui nécessite une grande fiabilité et un calcul en temps réel.[61]

Au total, la base de données se compose de 20 séquences de 60 images avec une fréquence d'images de 30 Hz. Tous les clips vidéo ont été capturés avec une position fixe de la caméra au milieu du toit du véhicule. Pour chaque séquence, nous fournissons une séquence d'annotation étiquetée manuellement, c'est-à-dire que toutes les images fournies sont annotées de manière dense.

La base de données Highway Driving fournit les contributions suivantes. Premièrement, nous fournit des clips vidéo réalistes sous un scénario de conduite. La conduite autonome est donc une application qui bénéficie directement des retombées de notre base de données. Deuxièmement, il fournit plus de 1 000 paires d'annotations de trame. Ils sont annotés et inspectés à plusieurs reprises par plusieurs annotateurs. Enfin, les annotations dans une

séquence reflètent la corrélation entre les trames adjacentes. Les images d'une séquence unique ont été annotées dans l'ordre chronologique, et les résultats anciennement annotés pour les images précédentes ont été fournis aux annotateurs comme référence. La figure 18 représente trois images de trois scènes différentes de la base de données choisie.

Figure 18 - Différentes scènes de la base de données Highway Driving.

3.3.2 Prétraitement : Fusionnement d'images successives

Le problème de la détection des voies de circulation peut être un problème difficile en raison de certaines conditions routières. Dans une séquence vidéo, parfois les marquages routiers blancs ne sont pas présents dans certaines scènes, ce qui les rend difficile à détecter ou totalement indétectables. Nous avons proposé un pré-traitement de fusionnement d'images successives qui s'applique à un ensemble d'images successives, son but est d'aider à relier les marquages de voies discontinues, et leur donnant l'apparence d'une ligne continue ou presque continue. La figure 20 représente le résultat de prétraitement dans deux scènes, l'image originale à gauche et la fusion de dix images successives à droite.

L'équation (4) utilisée pour fusionner les images successives :

$$I(k) = max\{I(k-1), I(k-2), \dots, I(k-p)\} \qquad (4)$$

Où I(k) est l'image à l'instant k et p est le nombre d'images successives à fusionner. Et ces deux algorithmes ci-dessous que nous avons proposé représente le fonctionnement de notre prétraitement telle que le premier algorithme prend en entrée un tableau d'images et fait la

comparaison entre les pixels de toutes les images et garde la valeur maximale et retourne en sortie une image qui représente le fusionnement de toute les images, et le deuxième algorithme prend un tableau d'images et applique le traitement du premier algorithme sur un ensemble de k images et retourne en sortie un tableau d'images fusionnées.

```
Fonction FusionnementPixelsImages(images) : //Fonction qui prend un tableau
d'images et renvoie le fusionnement de toutes les images.
ENTREE : tableau d'images
Variables : largeur, hauteur, i, j, pixel_map
SORTIE : Fusionnement de toute les images
DEBUT
    for i in range(largeur):
        for j in range(hauteur):
            r, g, b = (0,0,0) //init r, g, b à la valeur minimale.
            for image in images:
                r1, g1, b1= image.getpixel(i, j))//obtenir la valeur de pixel RVB de
l'image k et la comparer à l'image k-1.
                if(r1>r):
                    r = r1
                if(g1>g):
                    g = g1
                if(b1>b):
                    b = b1
    pixel_map[i, j] = (int(r), int(g), int(b))//réglage de la valeur du pixel.
FIN

Fonction FusionemntDesImages(images_entrée, k) :
//Fonction qui prend un tableau d'images en appliquant le fusionnement de k images.
ENTREE : tableau d'images, k
Variables : images_entrée, images_sortie, i, k
SORTIE : Fusionnement des k images
DEBUT
    While i<len(images_entrée):
        images_sortie.append(FusionnementPixelsImages(images_entrée[i-k:i]))
        i=i+1
FIN
```

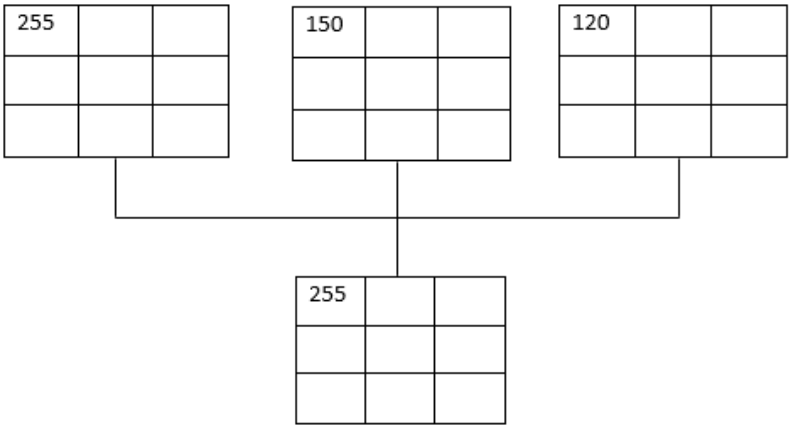

Figure 19 - Fonctionnement de prétraitement.

La figure 19 ci-dessus représente un exemple d'application des algorithmes précédents sur trois images successives. Le prétraitement compare les pixels des trois images et retourne la valeur maximale et nous le faisons sur tous les pixels restants des images.

Figure 20 - Résultats de la fusion d'images successives.

3.3.3 Extraction des caractéristiques

L'un des caractéristiques utiles et les plus efficaces dans la détection des objets dans les images est le contour. Le contour représente l'endroit où les valeurs des intensités des pixels successives dans une image changent très fortement, que nous appelons un gradient. Dans ce projet, un détecteur de contours Canny est proposé pour la détection des contours, et le résultat

se présente dans la figure 26. La figure 27 représente l'algorithme général de Détecteur de contour astucieux. Et qui suit seront les différentes étapes de l'algorithme Canny :

1) Convertir l'image en niveau de gris

Convertie l'image en niveau de gris comme le montre la figure 21, ce qui facilite la recherche d'informations et réduit le temps de calcul. Une image en niveaux de gris peut être obtenue à partir de l'espace colorimétrique RVB selon l'équation (5).

$$F(i,j) = 0.299R + 0.587V + 0.114B \qquad (5)$$

Figure 21 - Image en niveaux de gris.

2) Réduction de bruit par le filtre de gaussien

L'opérateur de lissage gaussien est un opérateur de convolution 2D qui est utilisé pour flouer les images et supprimer les détails et le bruit. Le filtre gaussien est un type de filtre qui utilise une fonction gaussienne pour calculer la transformation à appliquer à chaque pixel de l'image. La formule (6) est la fonction gaussienne en deux dimensions [66]. Et la figure 23 représente le résultat de ce filtre sur deux scènes.

$$G(i,j) = \frac{1}{2\pi\sigma^2} e^{-\frac{i+j^2}{2\sigma^2}} \qquad (6)$$

Où i est la distance à l'origine sur l'axe horizontal, j est la distance à l'origine sur l'axe vertical et σ est l'écart type de la distribution gaussienne.

Pour trouver les valeurs de filtre gaussien 3*3 il faut spécifier une valeur pour sigma σ et remplacer les valeurs de la matrice ci-dessous dans l'équation (6). Et la figure 22 représente un exemple de convolution d'une image avec un filtre gaussien.

G (-1, -1)	G (0, -1)	G (1, -1)
G (-1, 0)	G (0, 0)	G (1, 0)
G (-1, 1)	G (0, 1)	G (1, 1)

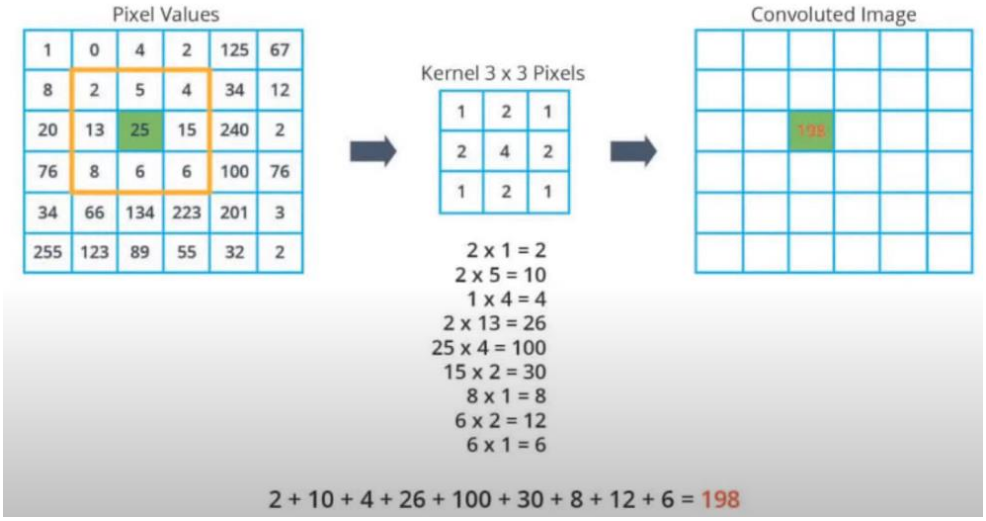

Figure 22 - Convolution d'une image avec un filtre de gaussien.

Figure 23 - Réduction du bruit par filtre de gaussien.

3) Trouver les gradients d'intensité de l'image.

Les filtres de Sobel sont convolués avec l'image pour trouver les gradients dans les directions i et j par la formule (1).

4) Suppression non maximum

L'algorithme passe par tous les points de la matrice d'intensité du gradient et trouve les pixels avec la valeur maximale dans les directions des contours qui sont calculées par la formule (3).

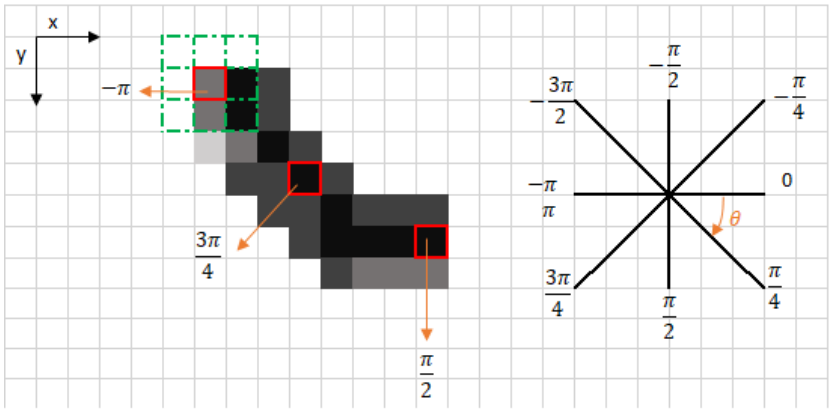

Figure 24 - Exemple de suppression non maximale [21]

Dans la figure 24 ci-dessus la case rouge du coin supérieur gauche représente un pixel d'intensité de la matrice d'intensité du gradient en cours de traitement. La direction de contour correspondante est représentée par la flèche orange.

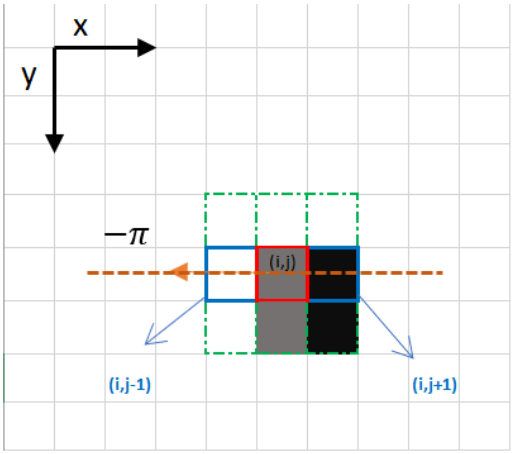

Figure 25 - suite de l'exemple de suppression non maximale [21]

Dans la figure 25, La direction du contour est la ligne pointillée orange (horizontale de droite à gauche). Le but de l'algorithme est de vérifier si les pixels d'une même direction sont plus ou moins intenses que ceux en cours de traitement. Dans l'exemple ci-dessus (figure 25), le pixel (i, j) est en cours de traitement et les pixels de la même direction sont surlignés en bleu (i, j-1) et (i, j+1). Si l'un de ces deux pixels est plus intense que celui en cours de traitement, alors seul le plus intense est conservé. Le pixel (i, j-1) semble être plus intense, car il est blanc (valeur de 255). Par conséquent, la valeur d'intensité du pixel actuel (i, j) est fixée à 0. S'il n'y a pas de pixels dans la direction des contours ayant des valeurs plus intenses, alors la valeur du pixel actuel est conservée. [21]

5) Double seuil

L'étape du double seuil vise à identifier trois types de pixels : forts, faibles et non pertinents :

1. Les pixels forts sont des pixels qui ont une intensité si élevée que nous sommes sûrs qu'ils contribuent au contour final.
2. Les pixels faibles sont des pixels qui ont une valeur d'intensité qui n'est pas suffisante pour être considérée comme forte, mais pas non pertinente.
3. Les autres pixels sont considérés comme non pertinents pour le contour.

- Principe de double seuil

1. Le seuil haut permet d'identifier les pixels forts (intensité supérieure au seuil haut)
2. Le seuil bas permet d'identifier les pixels non pertinents (intensité inférieure au seuil bas)
3. Tous les pixels ayant une intensité entre les deux seuils sont signalés comme faibles et le mécanisme d'hystérésis (étape suivante : Suivi des contours par hystérésis) nous aidera à identifier ceux qui pourraient être considérés comme forts et ceux qui sont considérés comme non pertinents. [68]

6) Suivi des contours par hystérésis

Sur la base des résultats de seuil, l'hystérésis consiste à transformer des pixels faibles en pixels forts, si et seulement si au moins un des pixels autour de celui en cours de traitement est fort. [68]

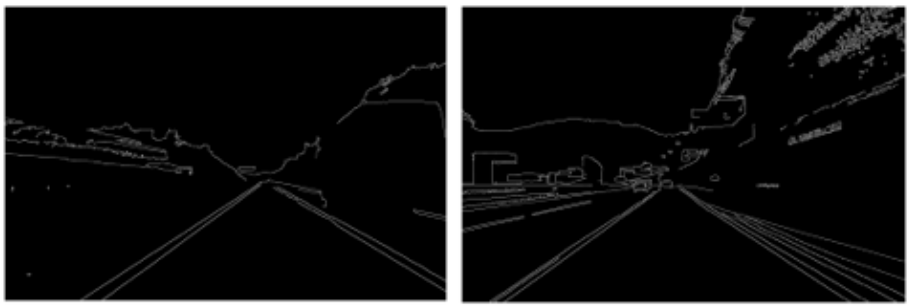

Figure 26 - Algorithme de Canny.

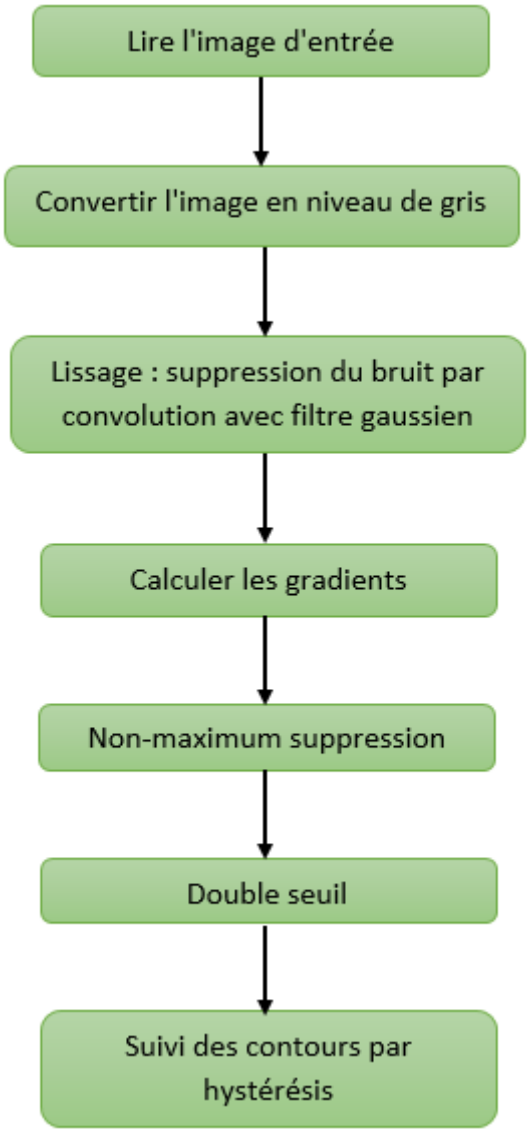

Figure 27 - Organigramme de l'algorithme général pour Détecteur de contour astucieux [18]

3.3.4 La région d'intérêt

Étant donné que la caméra est installée au milieu du toit de la voiture intelligente mais l'image de détection contenait des informations d'arrière-plan inutiles, telles que le ciel, les arbres, des deux côtés de la route. La partie de détection efficace, telle que les voies, peut représenter environ les deux tiers de la surface de l'image détectée appelée région d'intérêt (ROI), et l'extraction de ROI de l'image réduit les calculs inutiles et raccourcit le temps de consommation de l'étape de détection.

La région d'intérêt (ROI) est sélectionnée pour représenter un triangle qui ne contient que la partie des voies de circulation comme le montre la figure 29, ce qui les rend plus faciles à détecter. Nous représentons la région d'intérêt comme une image de masque binaire. Nous

avons créé un masque qui représente un tableau de zéros (image noire) et identique à la dimension de l'image traitée. Ensuite, nous remplissons la dimension du triangle dans ce masque avec l'intensité de 255 afin que les dimensions de notre région d'intérêt soient blanches. Enfin, nous appliquant le masque crée sur l'image de Canny et nous dessinons le triangle de la région d'intérêt par l'opérateur Bitwise-and. Les figure 28 et 29 montre avant et après la sélection de la région d'intérêt.

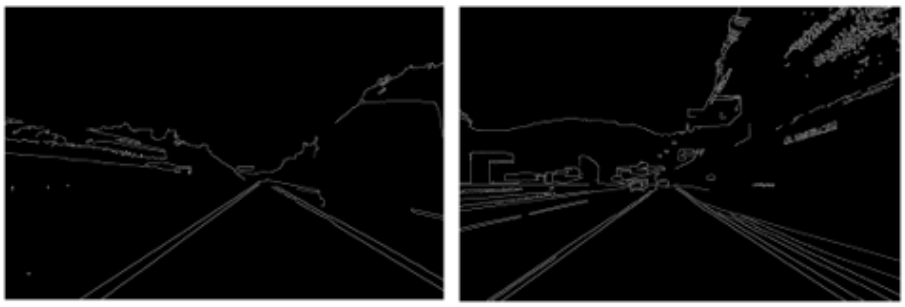

Figure 28 - Image avant la sélection de la région d'intérêt (ROI).

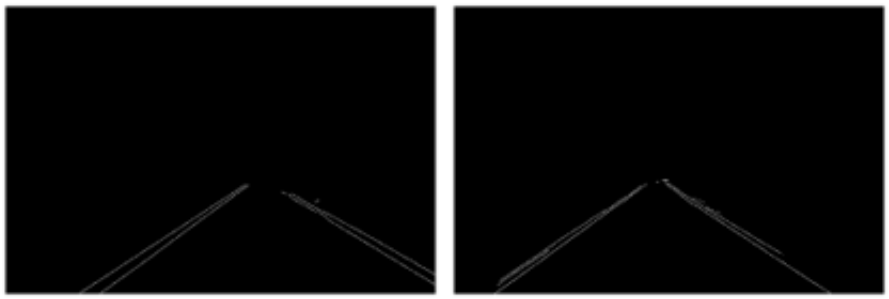

Figure 29 - Image après la sélection de la région d'intérêt (ROI).

3.3.5 Ajustement du modèle

1) Transformée de Hough probabiliste

La transformée de Hough probabiliste (PHT) est une technique de vision par ordinateur utilisée pour la détection de formes. Dans ce projet, nous avons utilisé PHT pour détecter des droites dans une image à partir des points alignés que nous avons obtenus grâce à la détection des contours. Car le but c'est obtenir des droites représentant les contours des formes. Et pour alléger la charge de calcul et de stockage, PHT n'utilise pas tous les points de contours M, seul un sous-ensemble m sélectionné au hasard est utilisé. Intuitivement, cela fonctionne car un sous-ensemble aléatoire de M représentera équitablement toutes les caractéristiques en fonction de la zone qu'ils occupent dans l'image. Le choix des points est basé sur un seuil fixe, seuls les

points qui ont des valeurs supérieures au seuil seront retournés. PHT effectue également des calculs de longueur de ligne, ce qui aide à détecter les segments. [67]

Cette méthode repose sur le paramétrage d'une droite par un angle θ et une distance ρ, l'équation paramétrique (7) de la droite est :

$$\rho = x \cos\theta + y \sin\theta \qquad (7)$$

Où, ρ représente le rayon depuis l'origine, θ représente l'angle de la normale à partir de l'origine, (x, y) représentent les coordonnées en pixels de la ligne. Cette équation définit la transformation entre l'espace (x, y) et l'espace (ρ, θ), illustrée à la figure 30 Par conséquent, toute ligne peut être transformée dans les termes de (ρ et θ).

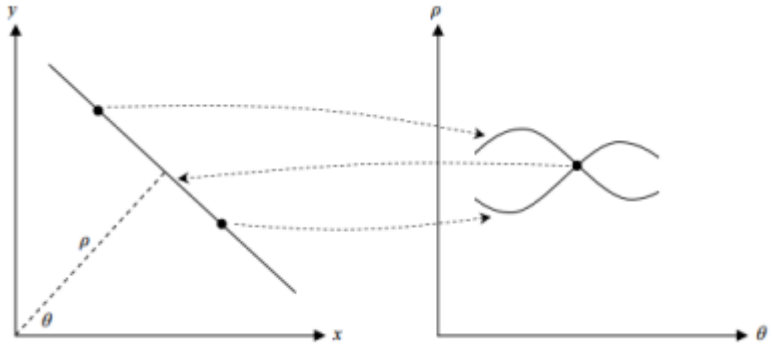

Figure 30 - L'espace (x, y) et l'espace (ρ, θ). [67]

Comme le montre la figure 30 les points dans l'espace image sont définis comme des courbes dans l'espace des paramètres, et l'intersection des courbes sinusoïdales est représentée par la ligne dans l'espace (x, y). La technique de transformation de Hough utilise un processus de vote en créant un accumulateur 2D qui se compose des deux paramètres (ρ et θ). La procédure itère sur tous les pixels de l'image et vote pour chaque cellule (ρ et θ) s'il y a suffisamment de preuves d'une ligne droite à ce pixel particulier. Après itération, les cellules avec les maximas locaux dans l'accumulateur sont très probablement des lignes droites, et la distance à l'origine sera le ρ voté, tandis que l'angle sera le θ.

Figure 31 - Voies détectées à l'aide de transformée de Hough probabiliste.

Le résultat de PHT apparaît sur la figure 31 dans deux scènes différentes.

2) Calcul de la moyenne et de l'extrapolation des lignes de voie

Le résultat de la transformée de Hough probabiliste se présente sous la forme de plusieurs lignes pour chaque voie. Nous avons fait la moyenne de ces plusieurs lignes et nous traçons une seule ligne pour chaque voie de gauche et de droite et nous devons également extrapoler les lignes pour couvrir toute la longueur de la ligne de voie. Dans cette étape, nous avons utilisé l'équation cartésienne (8) d'une ligne droite.

$$y = ax + b \qquad (8)$$

Tout d'abord, Nous trouvons les coordonnées a (la pente) et b (l'ordonnée à l'origine) des lignes obtenues par PHT de chaque image. Après, nous séparons les droites en deux groupes. La séparation se fait par la coordonnée a ('a' négative = ligne gauche, 'a' positive = ligne droite). Ensuite, nous calculons la moyenne des coordonnées a et b pour chaque groupe. Après cela, nous convertissons les coordonnées a et b de chaque ligne en points de pixel. Enfin, nous traçons des lignes pleine longueur à partir de points de pixel. La figure 32 représente l'organigramme générale de ce traitement. Le résultat de la moyenne et de l'extrapolation des lignes de voie est illustré à la figure 33 dans deux scènes différentes.

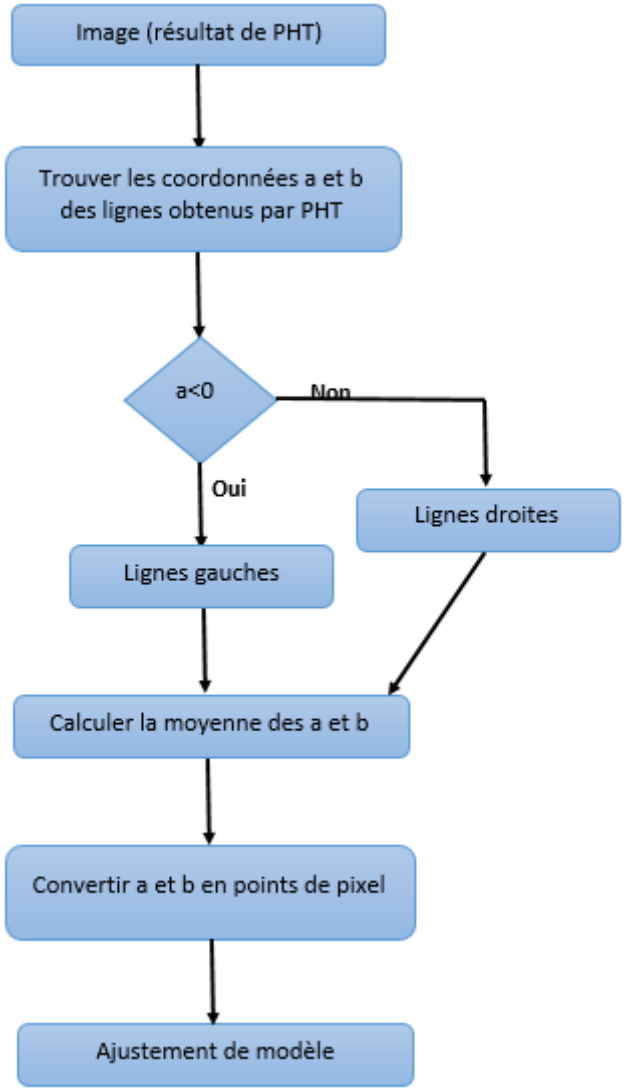

Figure 32 - Organigramme générale de la moyenne et extrapolation des lignes de voies.

Figure 33 - Moyenne et extrapolation des lignes de voies.

3.4 Ressources utilisées

3.4.1 Langage de programmation

Le langage de programmation choisi dans notre travail est le python. Python est un langage de programmation généraliste interprété de haut niveau. Actuellement, Python est considéré comme le langage de programmation le plus courant. Sa simplicité et sa flexibilité sont une partie des raisons pour lesquelles de nombreux développeurs de logiciels veulent l'utiliser. Indépendamment de sa simplicité, Python est un langage sur lequel vous pouvez compter pour effectuer des tâches complexes. Il a tendance à être utilisé pour faire de la préparation d'images et des faits saillants de reconnaissance. [64]

L'importance de Python l'a influencé pour être très ingénieux. Il est pressé avec quelques bibliothèques qui sont nécessaires pour quelques fonctionnalités, parmi lesquelles la reconnaissance d'image. L'une des bibliothèques les plus intenses et les plus efficaces est la « bibliothèque d'apprentissage automatique Scikit-Learn ». Outre la reconnaissance d'image, les bibliothèques peuvent être utilisées pour des capacités plus intelligentes, par exemple, confronter la reconnaissance et l'identification des mouvements.

Figure 34 – Logo de Python.

3.4.2 Google Colab

Google Colab a été développé par Google pour fournir un accès gratuit aux GPU (Processeur graphique) et aux TPU (Unité de traitement du tenseur) à quiconque en a besoin pour créer un modèle d'apprentissage automatique ou d'apprentissage en profondeur. Google Colab peut être défini comme une version améliorée de Jupyter Notebook. Jupyter Notebook est une application qui permet d'éditer et d'exécuter des documents Notebook via un navigateur Web ou un environnement de développement intégré (IDE). Au lieu de fichiers, vous travaillerez avec des Notebooks. Certaines des fonctionnalités les plus intéressantes sont répertoriées ci-dessous. [63]

- Tutoriels interactifs pour apprendre l'apprentissage automatique et les réseaux de neurones.
- Écrivez et exécutez du code Python 3 sans configuration locale.
- Exécutez les commandes du terminal à partir du Notebook.
- Importez des ensembles de données à partir de sources externes telles que Kaggle.
- Enregistrez vos blocs-notes sur Google Drive.
- Importez des blocs-notes depuis Google Drive.
- Service cloud gratuit, GPU et TPU.
- Intégrez avec les bibliothèques PyTorch, Tensor Flow, Open CV.

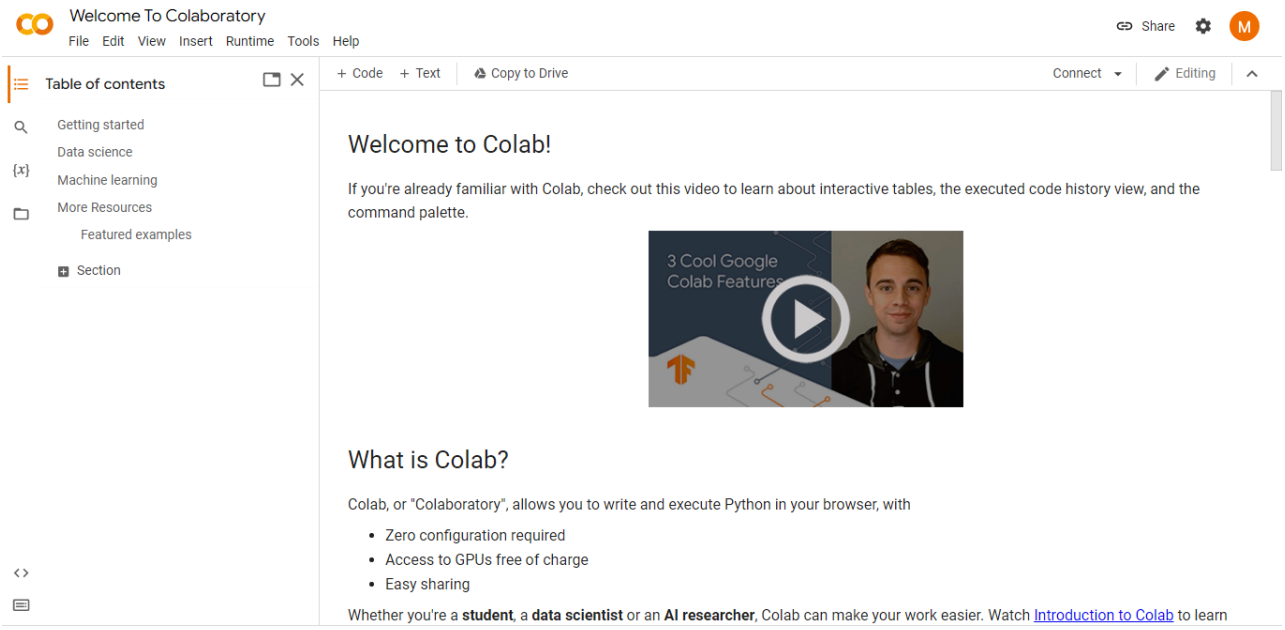

Figure 35 – Interface de Google Colab.

3.4.3 OpenCV

OpenCV (Open Source Computer Vision Library) la bibliothèque la plus célèbre pour la vision par ordinateur, l'apprentissage automatique et le traitement d'images et joue maintenant un rôle majeur dans le fonctionnement en temps réel, ce qui est très important dans les systèmes d'aujourd'hui. En l'utilisant, Nous pouvons traiter des images et des vidéos pour identifier des objets, des visages ou même l'écriture manuscrite d'un humain. Et dans notre projet, nous avons utilisé la version 4.4.5 de la bibliothèque OpenCV.[65]

Figure 36 - Logo de la bibliothèque OpenCV.

3.5 Résultat et comparaison

Les expérimentations sont réalisées sur la base de données Highway Driving, disponible dans [62].

Pour la comparaison, nous avons implémenté le détecteur de voie avec et sans le prétraitement de fusionnement de trames successives. Les résultats sont présentés dans la figure 37 qui montre le résultat dans quatre scènes différentes, détecteur de voie avec le prétraitement à droite et sans le prétraitement à gauche. Sans le prétraitement, dans la première scène nous avons marqué une fausse détection de la voie droite, la deuxième scène un manque de détection de la voie droite, la troisième scène un manque de détection de la voie gauche et pour la quatrième scène une fausse détection de la voie gauche, par contre avec le prétraitement les lignes sont orientées parfaitement dans les quatre scènes.

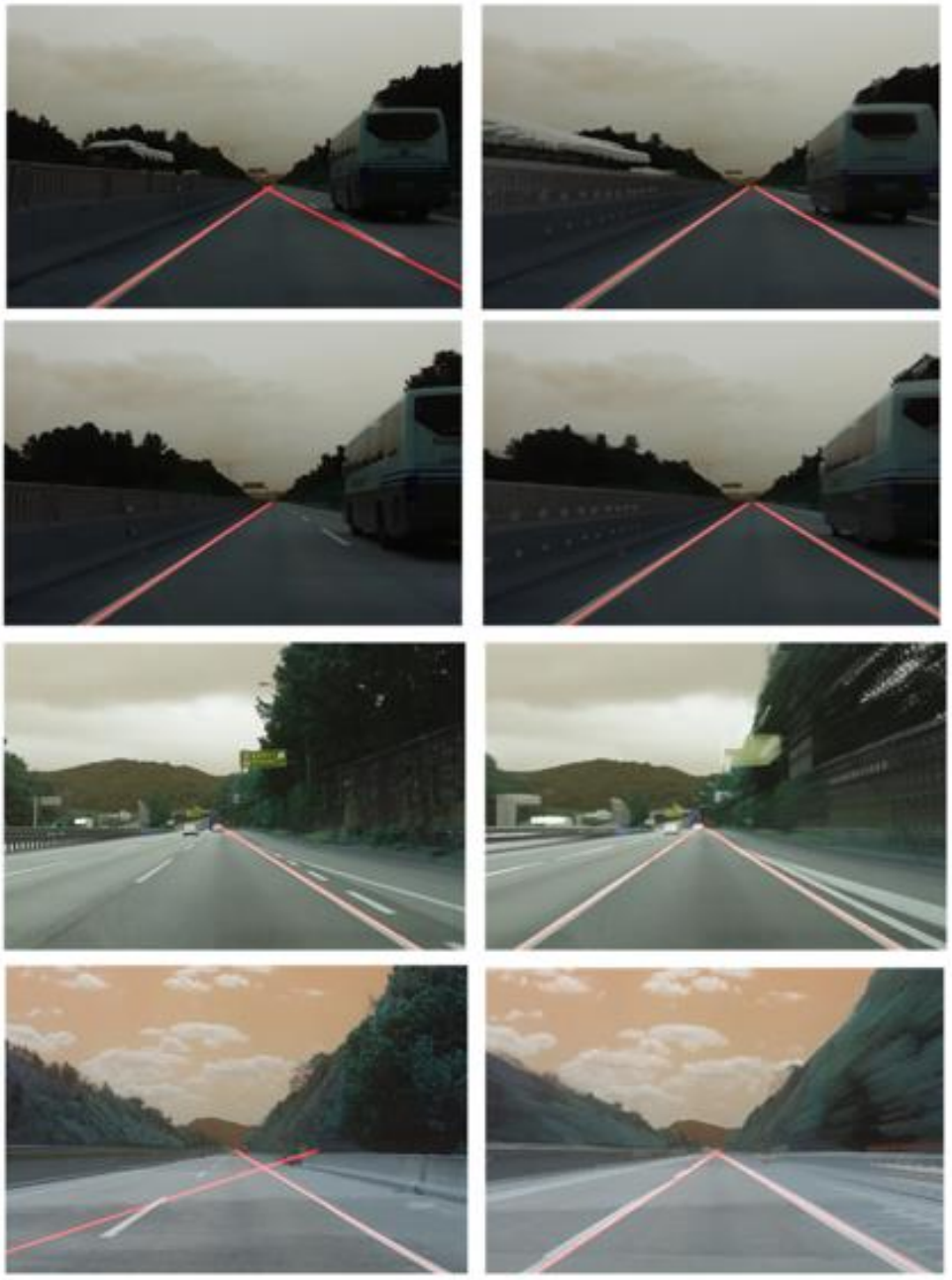

Figure 37 - Détecteur de voie avec et sans le fusionnement d'images successives.

3.5.1 Métriques d'évaluation

Pour évaluer la capacité de notre approche à effectuer la tâche de détection de voie sur la base de données de Highway Driving, les métriques d'évaluation sont utiles. Les tâches de détection de voie sont évaluées principalement à l'aide de métriques telles que la précision, Recall, F1-mesure, taux d'erreur et Accuracy. Ces mesures sont décrites ci-dessous en détail.

1) Précision

La précision est calculée comme le nombre de prédictions positives correctes divisées par le nombre total de prédictions positives. Précision montre combien de fausses détections dans notre modèle. S'il n'y a pas de fausses positives (FP), le modèle avait une précision à 100%, et elle est donnée par la formule (9).

$$Précision = \frac{TP}{TP+FP} \qquad (9)$$

Où TP (vrai positif) représente les voies détectées et FP (faux positif) représente les fausses détections.

2) Recall (rappel)

Dans le rappel, au lieu de compter le nombre de faux positifs prédits, le rappel examine le nombre de faux négatifs c'est à dire le nombre des voies indétectables, et il est défini par la formule (10).

$$Recall = \frac{TP}{TP+FN} \qquad (10)$$

Où TP représente les voies détectées et FN (vrai négatif) représente les voies indétectables.

3) F1-mesure

F1-mesure est la moyenne harmonique de précision et de Recall et il est donnée par la formule (11).

$$F1 - mesure = 2 * (\frac{Précision*Recall}{Précision+Recall}) \qquad (11)$$

4) Taux d'erreur (ERR)

Le taux d'erreur (ERR) est calculé comme le nombre de toutes les détections incorrectes divisé par le nombre total de base de données. Le meilleur taux d'erreur est de 0,0, tandis que le pire est de 1,0. ERR est calculé par la formule (12)

$$ERR = \frac{FP+FN}{P+N} \qquad (12)$$

Où FP représente les fausses détections, FN représente les voies indétectables, et P+N représente le nombre total des données.

5) Accuracy (taux de reconnaissance)

Cette mesure signifie le pourcentage des voies qui sont correctement détectées, et il est défini par la formule (13).

$$Accuracy = 1 - ERR \qquad (13)$$

Les résultats de ses métriques d'évaluation sont représentés dans le tableau 1 et les deux graphes dans les deux figures 38 et 39 pour notre méthode sans le prétraitement de fusionnement d'images successives, et les résultats avec le prétraitement sont représentés dans le tableau 2 et les deux graphes dans les deux figures 40 et 41.

Tableau 1 – Performances sur la base de données Highway-Driving sans le prétraitement.

Highway-Driving	**Précision**	**Recall**	**F1-mesure**	**Taux d'erreur**	**Accuracy**
Seq-test00	0,72	0,80	0,75	0,38	0,62
Seq-test01	0,70	0,88	0,77	0,35	0,65
Seq-train00	0,89	0,78	0,82	0,28	0,72
Seq-train03	0,65	1	0,78	0,35	0,65
Seq-train10	0,72	0,97	0,81	0,28	0,72
Seq-train13	0,85	1	0,91	0,15	0,85
Moyenne	**0,75**	**0,90**	**0,80**	**0,29**	**0,70**

Dans le tableau 1, la précision est 75% c'est-à-dire que 25% est considérée comme des fausses détections, Recall est de 90% c'est-à-dire que 10% des voies ne sont pas détectés, F1-mesure la moyenne harmonique de précision et Recall est de 80%, le taux d'erreur de la méthode sans le prétraitement est de 29%, et Accuracy est de 70% c'est-à-dire que 70% des voies sont correctement détectées. Ces résultats sont illustrés dans les figures 38 et 39 par des graphes.

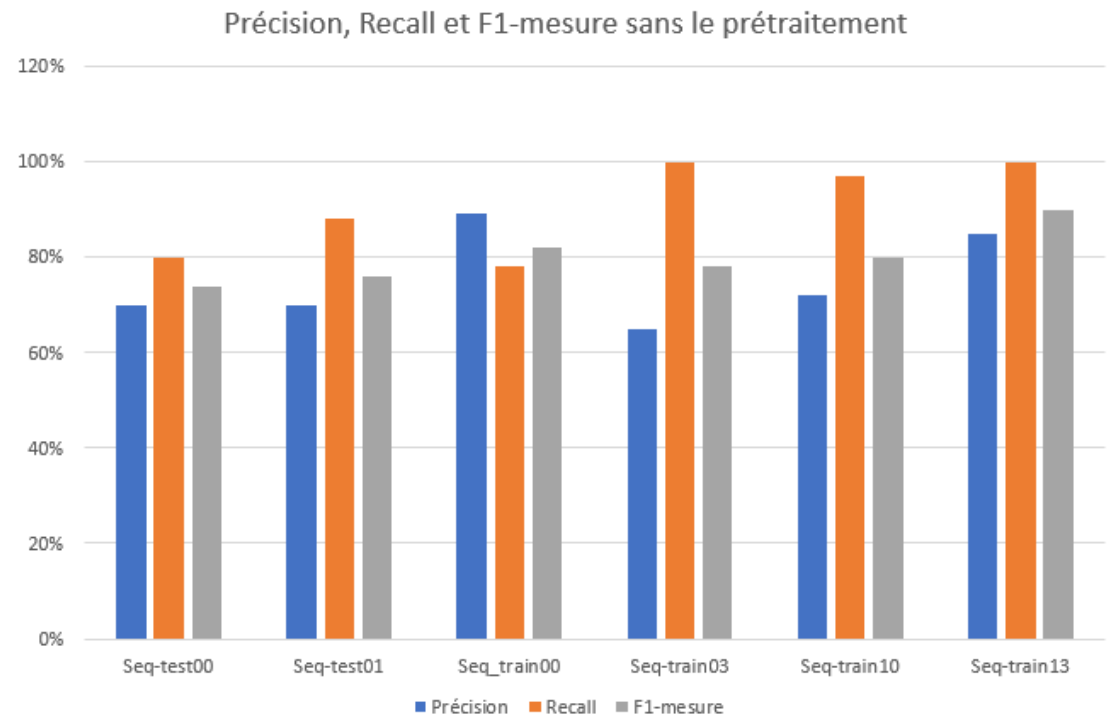

Figure 38 - Précision, Recall, et F1-mesure sans le prétraitement.

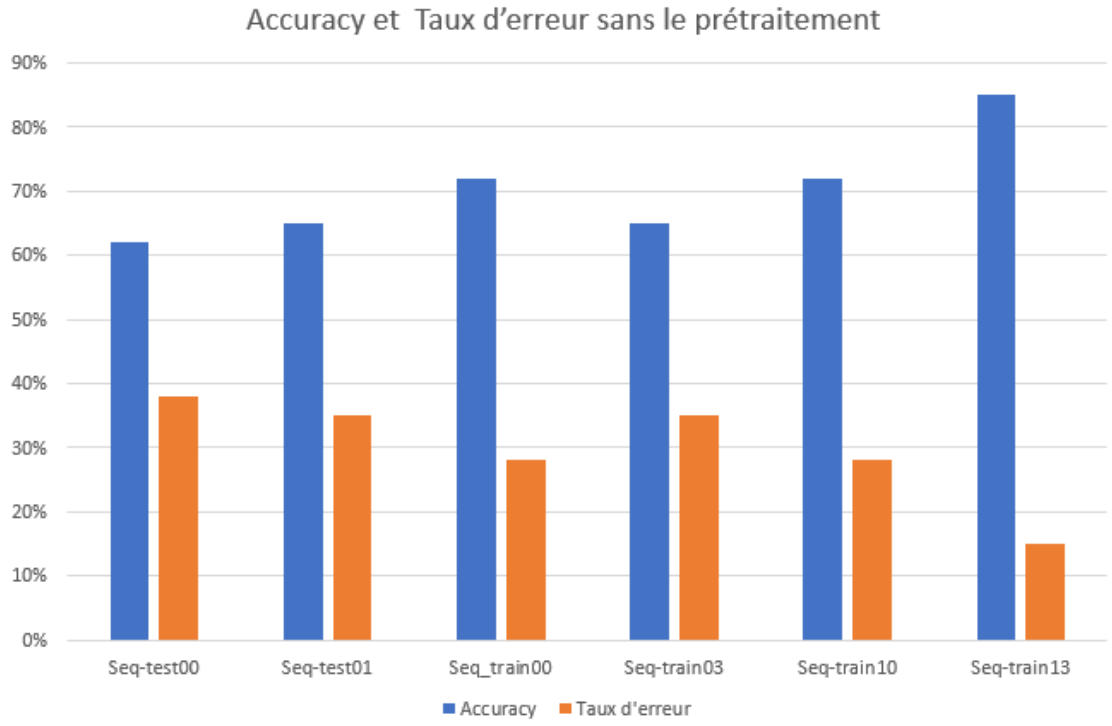

Figure 39 - Accuracy et taux d'erreur sans le prétraitement.

Tableau 2 – Performances sur la base de données Highway-Driving avec le prétraitement.

Highway Driving	Précision	Recall	F1-mesure	Taux d'erreur	Accuracy
Seq-test00	0,98	1	0,98	0,02	0,98
Seq-test01	0,91	1	0,95	0,08	0,92
Seq-train00	0,95	1	0,97	0,04	0,96
Seq-train03	0,89	1	0,94	0,08	0,92
Seq-train10	0,84	1	0,91	0,15	0,85
Seq-train13	0,85	1	0,91	0,15	0,85
Moyenne	**0,90**	**1**	**0,94**	**0,08**	**0,91**

Dans le tableau 2, la précision est de 90% c'est-à-dire que 10% est considérée comme des fausses détections donc nous avons marqué une amélioration de 15% par rapport à la méthode sans le prétraitement, Recall est de 100% et une amélioration de 10%, F1-mesure la moyenne harmonique de précision et Recall est de 94%, le taux d'erreur de la méthode avec le prétraitement est de 8% et une amélioration de 21%, et Accuracy 91% c'est-à-dire que 91% des voies sont correctement détectées et une amélioration de 21%. Ces résultats sont illustrés dans les figures 40 et 41 par des graphes.

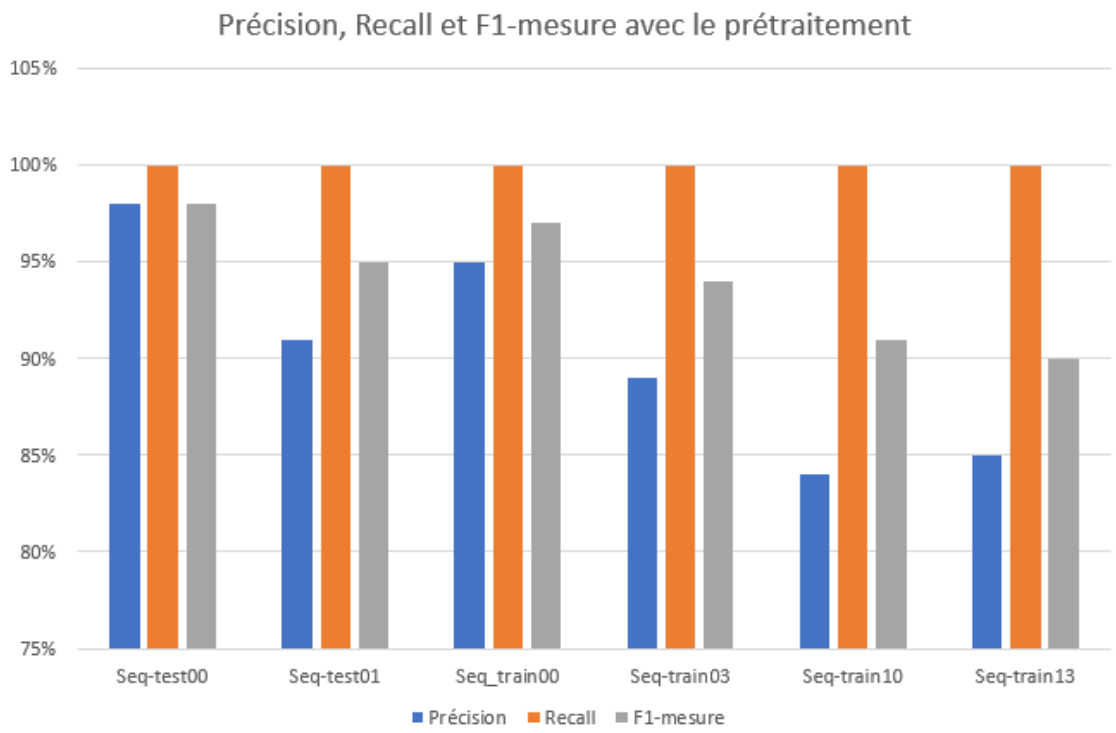

Figure 40 - Précision, Recall, et F1-mesure avec le prétraitement.

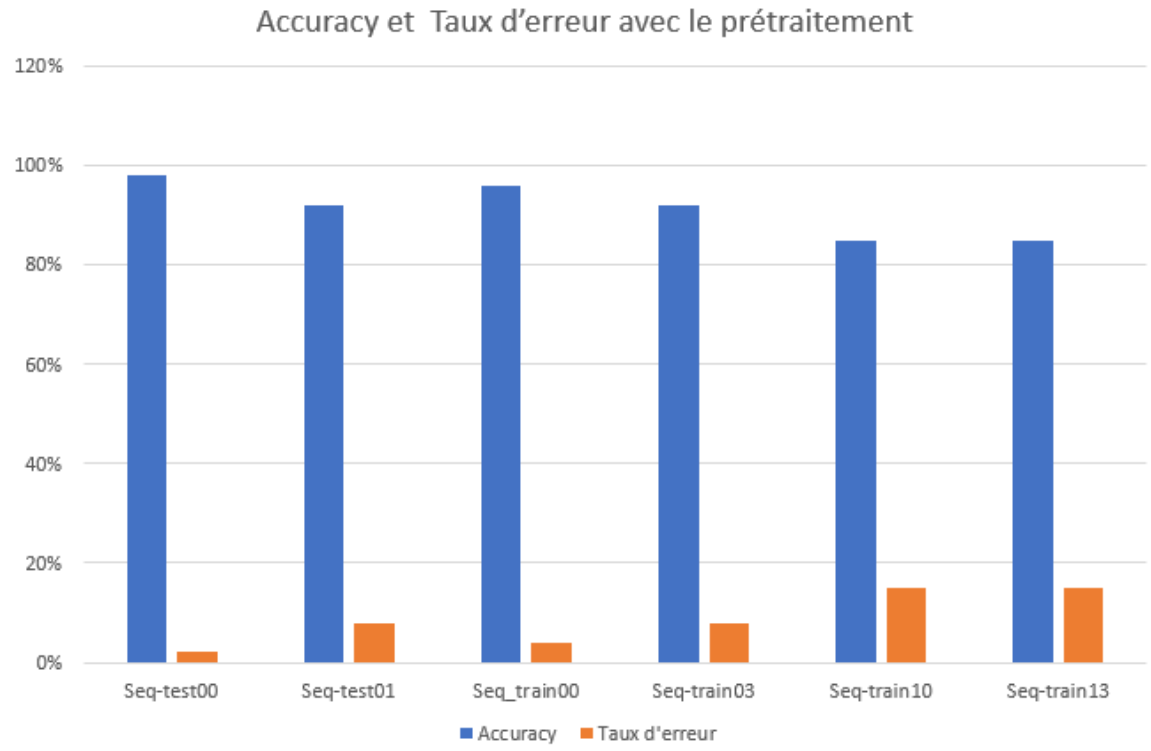

Figure 41 - Accuracy et taux d'erreur sans le prétraitement.

3.6 Conclusion

Dans ce chapitre, nous avons décrit les étapes de la réalisation de notre système de détection de la voie de roulement. Nous avons donné une importance aux méthodes de traitement d'images nécessaires pour notre algorithme puisqu'elles étaient notre sujet de travail, nous avons décrit l'architecture générale de l'algorithme adopté, nous avons détaillé bien la méthode de fusionnement des images successives comme étant un outil efficace pour améliorer la détection, les ressources utilisées, et le résultat de la comparaison entre le système sans et avec le prétraitement de fusionnement d'images successives.

Conclusion générale

Le développement des véhicules intelligents est actuellement le centre de nombreuses recherches sur les systèmes de transport pour mieux contrôler le véhicule et le comportement du conducteur considéré comme la cause de nombreux accidents. Pour cela, des systèmes d'aide à la conduite sont proposés, ayant pour objectif l'automatisation totale ou partielle de certaines tâches de la conduite. Parmi ces tâches, la détection des voies routiers.

L'objectif de développer des systèmes de détection et de suivi de voie est de fournir une fonction de sécurité et d'avertissement au conducteur pendant que les véhicules se déplacent sur la route.

Dans ce projet de fin d'étude, nous introduisons une méthode efficace de détection des lignes de route à l'aide de différentes images vidéo (base de de données Highway-Driving [62]) collectées dans le cadre du scénario de conduite. Pour cela nous nous intéressons aux techniques de traitement d'images et des algorithmes de détection des voies routières à l'aide de technologie de vision par ordinateur, notre travail était composé de quatre étapes, la première étape est le prétraitement de fusionnement d'images successives, la deuxième est l'extraction des caractéristiques en utilisant l'algorithme de Canny, la troisième est la région d'intérêt et la dernière étape est l'ajustement de modèle à 'aide de l'algorithme de la transformée de Hough probabiliste. Enfin, après une comparaison de notre approche proposée avec et sans le prétraitement de fusionnement d'images successives, le prétraitement aboutit à des résultats meilleurs.

Afin d'accomplir notre travail, nous avons commencé dans le premier chapitre par donner des généralités sur l'aide à la conduite et véhicule autonome et les concepts de base de notre étude où nous avons défini les techniques de traitement d'image.

Dans le deuxième chapitre, nous avons étudié les différents travaux des chercheurs dans le domaine de détection des voies routiers.

Dans le troisième chapitre, nous avons concentrer sur les différentes parties de modélisation et d'implémentation de notre méthode proposée pour le développement d'une application d'aide à la conduite qui se base sur la détection des voies routière basé sur le fusionnement d'images successives et nous avons obtenu un bon résultat et un bon taux de détection.

Lors de notre étude sur les algorithmes de détection, nous avons constaté que les recherches dans le domaine du véhicule intelligent sont nombreuses. En effet, elles ont donné lieu au développement d'une multitude de techniques allant de la détection des marquages de la voie et la localisation précise de l'automobile à la reconnaissance des panneaux routiers, détection de piétons, détection de véhicules, etc. Cependant, les solutions proposées jusqu'à maintenant ne sont pas toujours applicables dans des environnements non contrôlés ou non structurés et donc ce domaine connaitra encore d'autres avancé.

Bibliographie

[1] S. A. O. T. R. R. a. J. S. Jorge Vargas, «An Overview of Autonomous Vehicles Sensors and Their Vulnerability to Weather Conditions,» 10 August 2021.

[2] S. P. V. S. Dimil Jose, «Intelligent Vehicle Monitoring Using Global Positioning System and Cloud Computing,» *Procedia Computer Science,* vol. 50, pp. 440-446, 2015.

[3] A. B. H. ·. R. L. ·. D. L. ·. G. Raz, «Recent progress in road and lane detection: a survey,» *Machine Vision and Applications,* 7 February 2012.

[4] M. M. M. B. G. Marsden, «Towards an understanding of adaptive cruise control,» *Transportation Research Part C Emerging,* February 2001.

[5] H. C. Shinq-Jen Wu, «The Heterogeneous Systems Integration Design and Implementation for Lane Keeping on a Vehicle,» *IEEE Transactions on Intelligent Transportation Systems,* 1 June 2008.

[6] H. A. T. Gao, «Self lane assignment using egocentric smart mobile camera for intelligent GPS navigation,» *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops,* 20 June 2009.

[7] H. Halmaoui, «Restauration d'images par temps de brouillard et de pluie : applications aux aides à la conduite,» 30 novembre 2012.

[8] P. COHEN, «Where are the cameras in your car and what are they looking for?,» MAY 22, 2019. [En ligne]. Available: https://www.komando.com/tech-tips/where-are-the-cameras-in-your-car-and-what-are-they-looking-for/567371/.

[9] V. Tyagi, «Understanding Digital Image Processing,» *Department of Computer Science and EngineeringJaypee University of Engineering and TechnologyRaghogarh, Guna (MP), india,* September 2018.

[10] L. A. B. Nourria KEDDAR, «Détection et Reconnaissance Des Panneaux de signalisation Routière,» Aïn-Témouchent, 2019.

[11] D. Lamine, «Un système de détection des objets de la circulation routière et d'estimation de leur distance.,» BIKSRA, 2019-2020.

[12] B. ALDJIA, «Etude de l'effet des Transformées de Décorrélation en Compression des Images Couleurs RGB,» BATNA, 2010.

[13] D. B. FZ, «IMAGE NUMÉRIQUE,» Université Abdelhamid Ibn Badis Mostaganem Faculté des Sciences Exactes et Informatique Département des mathématiques et Informatique.

[14] J. J. Lizhe Tan, «Digital Signal Processing,» *Fundamentals and Applications Third Edition,* 2019.

[15] M. Naouai, «Filtrage d'image cours 7,» Universite de Tunis Elmanar, 2009-2010.

[16] L. C. Guang Deng, «An adaptive Gaussian filter for noise reduction and edge detection,» January 1993.

[17] J.-H. C. S.-B. C. Pham-Minh-Luan Nguyen, «An architecture for real-time hardware co-simulation of edge detection in image processing using Prewitt edge operator,» January 2014.

[18] M. K. a. R. S. Rashmi, «ALGORITHM AND TECHNIQUE ON VARIOUS EDGE DETECTION: A SURVEY,» *Signal & Image Processing : An International Journal (SIPIJ),* June 2013.

[19] J. C. Maxwell, «Digital Image Processing Mathematical and Computational Methods,» *Digital Image processing. Horwood Publishing,* 1868.

[20] J. F. Canny, «A Computational Approach To Edge Detection,» *IEEE Transactions on Pattern Analysis and Machine Intelligence,* December 1986.

[21] S. Sahir, «Canny Edge Detection Step by Step in Python — Computer Vision,» Jan 25, 2019. [En ligne]. Available: https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123.

[22] X. L. a. K. Otobe, «Hough transform algorithm for real-time pattern recognition using an artificial retina camera,» *Optics Express,* 2001.

[23] S. Bernard, «Extraction de primitives structurelles pour la reconnaissance de symboles : Une approche robuste par Transformée de Hough».

[24] A. S. Huang, D. C. Moore, M. Antone, E. B. Olson et S. Teller, «Finding Multiple Lanes in Urban Road Networks with Vision and Lidar,» *Autonomous Robots,* 2009-03.

[25] B.-S. J. Hsu-Yung Cheng, «Lane Detection With Moving Vehicles in the Traffic Scenes,» *IEEE Transactions on Intelligent Transportation Systems,* January 2007.

[26] l. Katramados, «Real-Time Traversable Surface Detection by Colour Space Fusion and Temporal Analysis,» October 2009.

[27] J. M. Á. M. L. Baldrich, «Shadow Resistant Road Segmentation from a Mobile Monocular System,» 2007.

[28] J. M. T. McCall, «Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation,» *IEEE Transactions on Intelligent Transportation Systems ,* 06 March 2006.

[29] K. W. A. N. T. N. Y. Yamaguchi, «Road region estimation using a sequence of monocular images,» 2008.

[30] C. B. Andrew Bacha, «Odin: Team VictorTango's entry in the DARPA Urban Challenge,» 23 July 2008.

[31] J. H. a. B. Marcotegui, «FILTERING OF ARTIFACTS AND PAVEMENT SEGMENTATION FROM MOBILE LIDAR DATA,» 2009.

[32] G. Zhang, « An efficient road detection method in noisy urban environment,» 2009.

[33] D. Pomerleau, «RALPH: rapidly adapting lateral position handler,» 1995.

[34] S. C. Alberto Broggi, «An agent based evolutionary approach to path detection for off-road vehicle guidance,» 2006.

[35] A. B. M. Bertozzi, «GOLD: a parallel real-time stereo vision system for generic obstacle and lane detection,» *IEEE Transactions on Image Processing,* 1998.

[36] B. M. E.D. Dickmanns, «Recursive 3-D road and relative ego-state recognition,» *IEEE Transactions on Pattern Analysis and Machine Intelligence.*

[37] K. K. Lakshmanan, «A deformable-template approach to lane detection,» *Intelligent Vehicles 95 Symposium.*

[38] S. Nedevschi et R. Schmidt, «3D lane detection system based on stereovision,» *Proceedings. The 7th International IEEE Conference on Intelligent Transportation Systems.*

[39] S. L. C. Kreucher, «LANA: a lane extraction algorithm that uses frequency domain features,» *IEEE Transactions on Robotics and Automation.*

[40] T. J. Dean Pomerleau, «Rapidly Adapting Machine Vision for Automated Vehicle Steering».

[41] M.-H. J. Dong-Joong Kang, «Road lane segmentation using dynamic programming for active safety vehicles,» 01 December 2003.

[42] A. Z. Nicholas Apostolof, «Robust Vision based Lane Tracking using Multiple Cues and Particle Filtering».

[43] W. K. Sukhan Lee, «Robust lane keeping from novel sensor fusion,» *IEEE International Conference on Robotics and Automation .*

[44] «Organisation mondiale de la Santé,» [En ligne]. Available: https://www.who.int/fr.

[45] Z. Kim, «Robust Lane Detection and Tracking in Challenging Scenarios,» *IEEE Transactions on Intelligent Transportation Systems,* 2008.

[46] E. K. T. D. S. Yue Wanga, «Lane detection and tracking using B-Snake,» *Image and Vision Computing,* 2003.

[47] L. S. M. Nieto, «Robust multiple lane road modeling based on perspective analysis,» *IEEE International Conference on Image Processing,* 2008.

[48] G. Y. E. D. SIBEL YENIKAYA, «Keeping the Vehicle on the Road – A Survey on On-Road Lane Detection Systems,» *ACM Computing Surveys,* 2013.

[49] C.-Y. W. Shih-Hsuan Chiu, «A FAST RANDOMIZED GENERALIZED HOUGH TRANSFORM FOR ARBITRARY SHAPE DETECTION,» *International Journal of Innovative Computing,* January 2011.

[50] Z. W. D. M. Jie Guo, «Lane Detection Method Based on Improved RANSAC Algorithm,» *IEEE Twelfth International Symposium on Autonomous Decentralized Systems,* 2015.

[51] K. P. S. King Hann Lim, «River Flow Lane Detection and Kalman Filtering-Based B-Spline Lane Tracking,» *International Journal of Vehicular Technology,* 05 Nov 2012.

[52] L. L. D. H. Jinyu Liu, «Lane Detection Based on Straight Line Model and K-Means Clustering,» *IEEE 7th Data Driven Control and Learning Systems Conference,* 2018.

[53] X. W. H. H. Huarong Xu, «A fast and stable lane detection method based on B-spline curve,» China, 2009.

[54] D. Schwartz, «Clothoid road geometry unsuitable for sensor fusion clothoid parameter sloshing,» *IEEE IV2003 Intelligent Vehicles Symposium. Proceedings,* 2003.

[55] A. B. A. F. Massimo Bertozzi, «Stereo inverse perspective mapping: theory and applications,» *Image and Vision Computing* .

[56] M. Benkedadra, «Image Enhancement Based On Supervised Learning,» MOSTAGANEM, 2019 - 2020.

[57] V. V. K. B. P. S. B. J. J. H. Bottazzi, «Adaptive regions of interest based on HSV histograms for lane marks detection,» *Robot Intelligence Technology and Applications 2,* 2014.

[58] B. M. Hamdi cherif Fethi, «Détection de la voie de roulement,» MOSTAGANEM, 2015/2016 .

[59] B. Nassima, «DETECTION DES BORDS DE ROUTES «Application De La Transformée De Radon »,» Oran, 15 décembre 2013.

[60] D. P. Mrs. P.M.Daigavane, «Road Lane Detection with Improved Canny Edges Using Ant Colony Optimization,» *Third International Conference on Emerging Trends in Engineering and Technology,* 2010.

[61] J. H. Z. H. W. W. Gen YANG, «A new hough transform operated in a bounded cartesian coordinate parameter space,» *IET Image Processing,* vol. 16, pp. 2282-2295, 2022.

[62] J. Y. J. K. Byungju Kim, «Highway Driving Dataset for Semantic Video Segmentation,» 2018. [En ligne]. Available: https://sites.google.com/site/highwaydrivingdataset/.

[63] H. Mane, «What is Google Colab?,» 28 Apr 2019. [En ligne]. Available: https://medium.com/@iHrishi_mane/what-is-google-colab-eb1e718646ce.

[64] «What is Python?,» 2020. [En ligne]. Available: https://www.teradata.com/Glossary/What-is-Python .

[65] K. Kargın, «Computer Vision Fundamentals and OpenCV Overview,» 4 May 2021. [En ligne]. Available: https://medium.com/mlearning-ai/computer-vision-fundamentals-and-opencv-overview-9a30fe94f0ce.

[66] I. Y. ,. a. Y. K. Yudai Yamaguchi, «Proposal of Edge-Preserving, Image Noise Reduction Filter for Using L2-Norm †,» *Engineering Proceedings,* vol. 11, p. 27, 2021.

[67] D. P. a. J. Rask, «A model based approach to lane detection and lane positioning using OpenCV,» CHALMERS UNIVERSITY OF TECHNOLOGY AND UNIVERSITY OF GOTHENBURG, Gothenburg, Sweden, 2017.

[68] A. S. H. K. A.A. Fallah, «Real-time Lane Detection Based on Image Edge Feature and Hough Transform,» *Journal of Electrical and Computer Engineering Innovations,* vol. 9, pp. 193-202, 2021.