République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
UNIVERSITE IBN KHALDOUN - TIARET

MÉMOIRE

Présenté a :

FACULTE DES MATHEMATIQUES ET INFORMATIQUE
DEPARTEMENT D'INFORMATIQUE

Pour obtenir diplome de :

MASTER

Spécialité : *Génie Logiciel*

YASSINE BOUAZA

Thème

Détection des plaques d'immatriculation et amélioration de la reconnaissance de leurs caractères (chiffres ou lettres)

Soutenu publiquement le 11/07/2023 à Tiaret devant le jury composé de :

• Mr. Baghdadi Mohamed	• MCB	• Président
• Mr. Gafour Yacine	• MCB	• Encadreur
• Mr. Mezzoug Karim	• MAA	• Examinateur

2022/2023

Dédicace

“

À mes très chers parents,

À mes chers frères et soeurs,

À mes chers amis,

À tous ceux qui me sont chers, et proches

Merci.

”

Yassine Bouaza

Remerciements

Tout d'abord, je remercie Allah le tout puissant de m'avoir donné le courage et la patience nécessaires à mener ce travail à son terme.

وَمَا تَوْفِيقِي إِلَّا بِاللَّهِ ۚ عَلَيْهِ تَوَكَّلْتُ وَإِلَيْهِ أُنِيبُ

Je tiens à remercier tout particulièrement mon encadrant **M. GAFOUR Yacine**, pour l'aide compétente qu'il m'a apporté, pour sa patience et son encouragement. Son œil critique m'a été très précieux pour structurer le travail et pour améliorer la qualité des différentes sections.

Un très grand remerciement et une très grande reconnaissance sont destinés à **M. RAHMOUNE Karim** de m'avoir aidé à obtenir ce stage de fin d'études chez Naftal CBR.

Je désire remercier également le chef de département de l'Informatique **M. MERATI Medjeded** pour leurs encouragements.

Que les membres de jury trouvent, ici, l'expression de mes sincères remerciements pour l'honneur qu'ils me font en prenant le temps de lire et d'évaluer ce travail.

Je souhaite aussi remercier l'équipe pédagogique et administrative de l'université pour leurs efforts dans le but de nos offrir une excellente formation.

Pour finir, je souhaite remercier toute personne ayant contribué de prés ou de loin à la réalisation de ce travail.

Résumé

Le contrôle du trafic et l'identification des propriétaires de véhicules est un problème majeur dans tous les pays. Il peut être difficile d'identifier le propriétaire de véhicule qui viole le code de la route et conduit trop vite. Ces systèmes sont basés sur différentes méthodologies, mais c'est toujours une tâche très difficile que certains facteurs tels que la plaque d'immatriculation non uniforme du véhicule, la vitesse élevée du véhicule, les différentes conditions d'éclairage, la météo, la phénomène de mirage et la langue du numéro de véhicule peuvent affecter beaucoup le taux de reconnaissance global. Dans ce travail, nous allons présenter un projet qui se base sur l'apprentissage automatique pour détecter de plaques d'immatriculation, et en utilisant une segmentation appropriée afin d'améliorer la reconnaissance de leurs caractères (chiffres ou lettres).

Mots clés : Reconnaissance automatique de plaques d'immatriculation (ANPR), apprentissage automatique, segmentation de caractères, Reconnaissance optique de caractères.

Abstract

Traffic control and identification of vehicle owners is a major problem in all countries. It can be difficult to identify the vehicle owner who violates the rules of the road and drives too fast. These systems are based on different methodologies, but it is still a very difficult task that some factors such as non-uniform vehicle license plate, high vehicle speed, different lighting conditions, the weather, mirage phenomenon and language of vehicle number vehicle can greatly affect the overall recognition rate. In this work, we will present a project based on machine learning to detect license plates, and using an appropriate segmentation to improve the recognition of their characters (numbers or letters).

Keywords : Automatic number-plate recognition (ANPR), Machine learning, Classification, Character Segmentation, Optical Character Recognition.

ملخص

يعد التحكم في حركة المرور وتحديد أصحاب المركبات مشكلة رئيسية في جميع البلدان. قد يكون من الصعب تحديد مالك السيارة الذي ينتهك قواعد الطريق ويسير بسرعة كبيرة. تعتمد هذه الأنظمة على منهجيات مختلفة ، ولكن لا يزال من الصعب جدًا أن تؤثر بعض العوامل مثل لوحة ترخيص المركبة غير الموحدة ، وسرعة السيارة العالية ، وظروف الإضاءة المختلفة ، الأحوال الجوية ، ظاهرة السراب ولغة رقم السيارة بشكل كبير على معدل التعرف الإجمالي. في هذا العمل ، سنقدم مشروعًا يعتمد على التعلم الآلي لاكتشاف لوحات الترخيص ، واستخدام التقسيم المناسب لتحسين التعرف على أحرفها (أرقام أو أحرف).

كلمات مفتاحية :

التعرف التلقائي على لوحات المركبات, تعلم آلي, تجزئة الحروف, التعرف البصري على الحروف.

Table des matières

Table des figures

[1] *Open Source Computer Vision*

Liste des tableaux

Liste des sigles et acronymes

ADAS	*Advanced Driver Assistance Systems*
RGB	*Red Green Blue*
CMYK	*Cyan Magenta Yellow and Key*
GIF	*Graphics Interchange Format*
JPEG	*Joint Photographic Experts Group*
TIFF	*Tag Image File Format*
PS	*Photoshop*
PSD	*Photoshop Document*
CNN	*Convolutional Neural Network*
YOLO	*You Only Look Once*
SOTA	*State-of-the-Art*
COCO	*Common Objects in Context*
OpenCV	*Open Source Computer Vision*
IA	*Intelligence Artificielle*
SVM	*Support Vector Machines*
ROI	*Region Of Interest*
OCR	*Optical Character Recognition*

Introduction générale

Modèle utilisé

Dans le système de transport intelligent, l'une des choses les plus importantes est d'identifier les les véhicules qui circulent sur la route. L'acquisition vidéo à l'aide de caméras de surveillance déjà installées sur l'autoroute est le moyen le plus simple à mettre en œuvre. Une plaque d'immatriculation est l'un des types d'identification des véhicules à moteur. Les plaques d'immatriculation sont également appelées plaques d'immatriculation des véhicules. Il s'agit d'une pièce de métal ou de plastique attachée à un véhicule à moteur en tant qu'identification officielle. Généralement, une paire de plaques d'immatriculation doit être installée à l'avant et à l'arrière du véhicule. Diverses méthodes d'identification des plaques d'immatriculation des véhicules ont été mises en œuvre. En général, ces algorithmes sont développés en 3 étapes, à savoir la recherche de la zone de la plaque d'immatriculation, la segmentation des caractères de la plaque d'immatriculation et la reconnaissance de chaque caractère. Le système de reconnaissance des plaques d'immatriculation des véhicules est une application qui remplace la fonction de la vision humaine dans la reconnaissance des plaques d'immatriculation des véhicules à moteur.

Problématique et contributions du mémoire

La lecture automatique des plaques d'immatriculation est une composante logicielle qui repose sur un ensemble d'algorithmes de reconnaissance d'images et de caractères. Ces algorithmes sont de nature complexe et délicate à implémenter.

La vraie problématique dans ce domaine réside dans la partie de segmentation des caractères afin d'appliquer la reconnaissance optique des caractères sur une petite partie d'image extraite qui contient des séquences de caractères. Ceci est souvent dans des conditions de grande vitesse et de faible luminosité. De plus, le fait de ne disposer que de très peu d'images hautes définition par seconde sur la plupart des caméras vidéo entraîne un manque de netteté lors de la capture de ces images. Pour cela il faut procéder à la reconnaissance des caractères qui s'apparente plus particulièrement à de l'analyse, de la manipulation, de l'amélioration d'images, de la détection de contour, pour permettre la reconnaissance optique des caractères (OCR[2]). [1]

En utilisant les fonctionnalités d'OpenCV et en intégrant une nouvelle méthode de

[2] *Optical Character Recognition*

segmentation, notre objectif est d'améliorer la précision et l'efficacité de la capture et de l'identification des numéros de plaque des véhicules.

Organisation du mémoire

Ce mémoire est organisé en trois chapitres :

Ce mémoire est commencé par une "**Introduction générale**" où J'expose quelques notions de bases sur le traitement d'images.

Le premier chapitre "**Généralité sur les images numériques**". Dans ce chapitre, j'ai exposé quelques notions de bases sur le traitement d'images.

Le deuxième chapitre "**Extraction de caractéristiques pour la reconnaissance des objets**" Ce chapitre fournit des connaissances introductives sur la vision par ordinateur que permet d'extraire, d'analyser et de comprendre automatiquement les informations utiles à partir d'une image.

Le troisième chapitre "**Segmentation pour améliorer la reconnaissance des caractères d'une plaque d'immatriculation**" Dans ce chapitre, nous donnons un aperçu des méthodes et des algorithmes proposés pour représenter les images par les chercheurs.

On ce termine par "**Conclusion générale**" Présente le résume de les contributions de recherche et les suggestions de recherche possibles pour les travaux futurs.

Chapitre 1

Généralité sur les images numériques

1.1 Introduction

L'étude et la manipulation des images numériques est utilisée pour améliorer leur qualité en extrayant des informations pertinentes. cette étude est connue regroupées sous le terme de traitement d'images. Ce domaine d'étude, présent à la fois en informatique et en mathématiques appliquées qui englobe un ensemble de procédés et de techniques visant à améliorer l'apparence visuelle des images.

Dans ce chapitre, nous examinerons certaines définitions fondamentales qui faciliteront la compréhension des méthodes d'analyse d'images. Nous aborderons également des caractéristiques clés à prendre en compte dans ce domaine.

1.2 Définition une image

Une image peut être considérée comme un signal 2D (x, y) qui représente souvent une réalité tridimensionnelle (x, y, z). Il y a deux perspectives principales sur la notion d'image [2].

- Du point de vue humain : Une image contient plusieurs informations sémantiques qui nécessitent une interprétation au-delà des valeurs numériques. Les éléments visuels de l'image sont interprétés pour comprendre le contenu et la signification.
- Du point de vue mathématique : Une image est une matrice de nombres qui représente un signal. Différents outils mathématiques peuvent être utilisés pour manipuler ce signal et extraire des informations.

Il existe trois principaux types d'images :

- Images en niveaux de gris : Les valeurs des pixels dans ces images appartiennent à l'intervalle [0, 255]. Elles représentent différentes nuances de gris, où 0 correspond au noir et 255 correspond au blanc.
- Images binaires : Ces images sont composées uniquement de deux couleurs, le noir et le blanc. Les pixels ne peuvent prendre que les valeurs 0 ou 1. Habituellement, le 0 représente le noir ou l'absence de contenu, tandis que le 1 représente le blanc ou la présence de contenu.
- Images en couleur : Ces images utilisent un mélange des trois couleurs primaires (rouge, vert, bleu) pour représenter une large gamme de couleurs. Chaque canal de couleur (rouge, vert, bleu) peut prendre des valeurs dans l'intervalle [0, 255] pour déterminer l'intensité de cette couleur à un pixel donné.

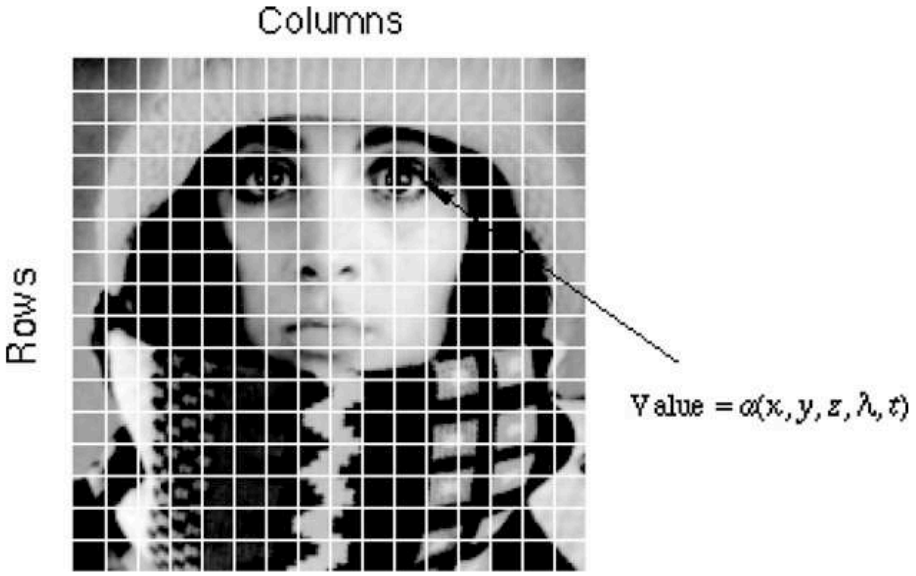

FIG. 1.1 : Numérisation d'une image continue. Le pixel aux coordonnées [m=10, n=3] a la valeur de luminosité entière 110.

1.3 Qu'est-ce qu'une image ?

Une image est une matrice, de pixels disposés en colonnes et lignes.

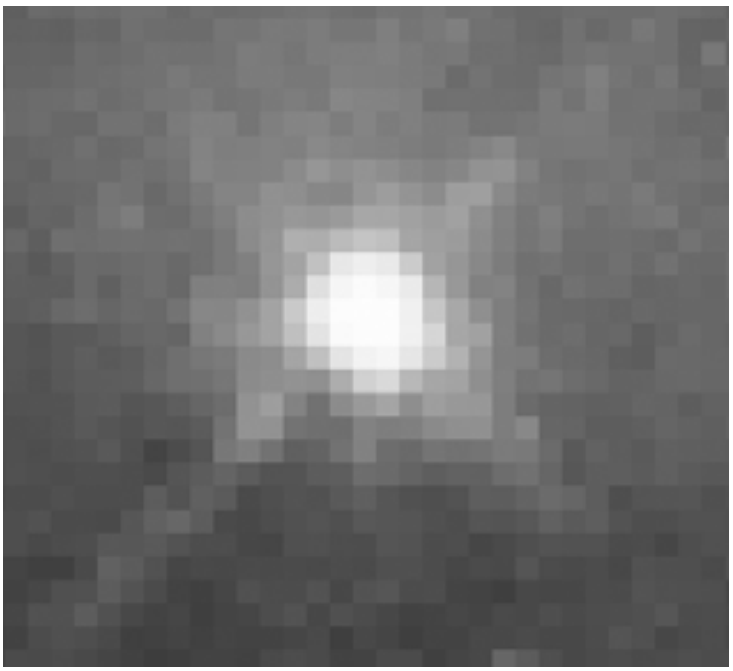

FIG. 1.2 : Une image - une matrice de pixels disposés en colonnes et en lignes.

Dans une image en niveaux de gris (8 bits), chaque élément d'image a une intensité assignée. La plage de valeurs est de 0 à 255. Une image en niveaux de gris est ce que l'on appelle communément une image en noir et blanc.

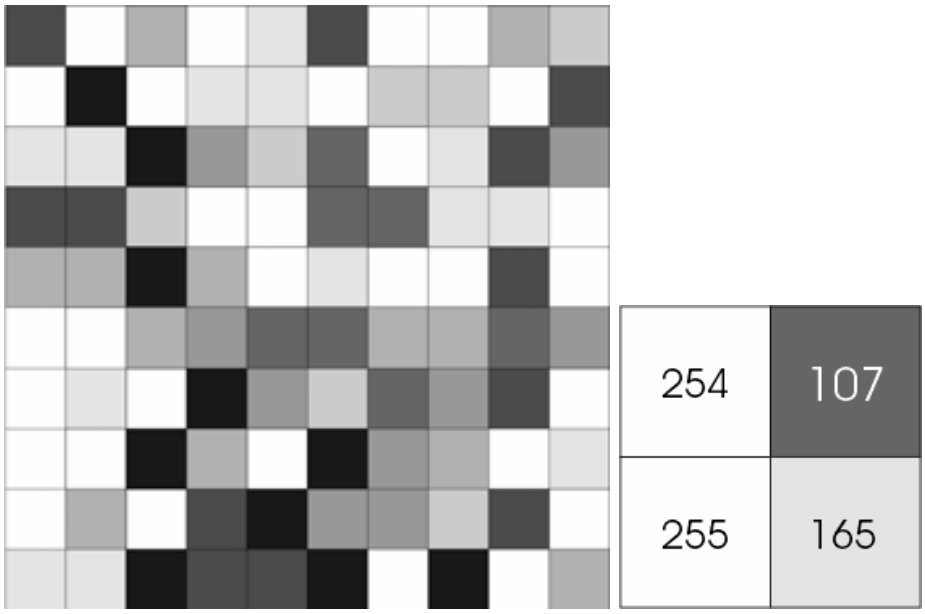

FIG. 1.3 : Chaque pixel a une valeur comprise entre 0 (noir) et 255 (blanc). La plage possible des valeurs de pixel dépend de la profondeur de couleur de l'image, ici 8 bits = 256 tons ou niveaux de gris.

Une image en niveaux de gris normale a une profondeur de couleur de 8 bits (256 values en niveaux de gris). L'image a une profondeur de couleur de 24 bits = 8 x 8 x 8 bits = 256 x 256 x 256 couleurs = 16 Des milliers de couleurs qui represente une image.

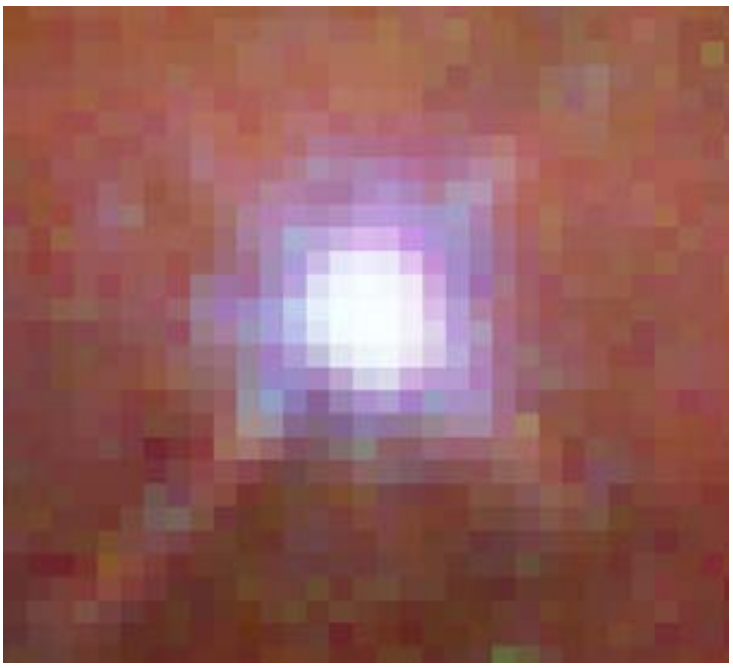

FIG. 1.4 : Une image en couleurs à partir de trois images en rouge, vert et bleu.

1.4 Échantillonnage et quantification

L'échantillonnage est un processus de dévision d'un signal analogique à intervalles réguliers pour créer un signal numérique discret. La quantification consiste à attribuer à chaque échantillon une valeur numérique représentant l'amplitude du signal analogique à un moment précis. L'échantillonnage et la quantification sont utilisés ensemble pour convertir un signal analogique continu en un signal numérique discret.

La conversion analogique-numérique est un processus réversible, qui permet de reconstruire un signal analogique à partir d'un signal numérique. Cependant, de telles reconstructions sont limitées, notamment en ce qui concerne la bande passante et les distorsions induites par la quantification.

Les techniques d'échantillonnage et de quantification sont utilisées dans de nombreuses applications, notamment l'enregistrement audio, la vidéo, la mesure et le contrôle de processus, la télémétrie, les communications sans fil, etc.

L'échantillonnage et la quantification peuvent également être utilisés dans le traitement du signal numérique pour effectuer des opérations telles que le filtrage, la compression, la modulation et la démodulation.

1.5 Espace de couleur de représentation des images

Les deux principaux espaces colorimétriques sont utilisés RGB,CMYK et Gamut.

1.5.1 RGB

L'espaces de couleur RGB[1] est très étroitement lié à la façon dont nous percevons la couleur (Rouge, Vert et Bleu) dans nos rétines. RGB utilise un mélange de couleurs additif et est la base modèle de couleur utilisé à la télévision ou dans tout autre média qui projette de la couleur avec de la lumière. C'est le modèle de couleur de base utilisé dans les ordinateurs et pour les graphiques Web, mais il ne peut pas être utilisé pour la production d'impression. Les couleurs secondaires de RGB - cyan, magenta et jaune - sont formées en mélangeant deux des couleurs primaires (rouge, vert ou bleu) et à l'exclusion de la troisième couleur. Rouge et le vert se combinent pour faire du jaune, du vert et du bleu pour faire du cyan, et du bleu et du rouge forme magenta. La combinaison du rouge, du vert et du bleu en pleine intensité donne du blanc.

[1] *Red Green Blue*

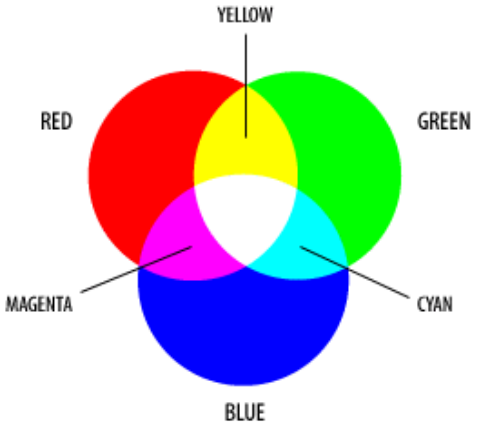

FIG. 1.5 : Le modèle de couleur RGB

1.5.2 CMYK

Le modèle de couleur CMYK[2] est utilisé dans l'impression pour reproduire une large gamme de couleurs. En mélangeant les couleurs primaires cyan, magenta et jaune, ainsi que le noir, on peut obtenir différentes teintes. Cependant, les couleurs CMY seules ne peuvent pas créer un vrai noir, mais seulement une couleur marron foncé. C'est pourquoi le noir est ajouté comme quatrième couleur d'impression dans le modèle CMYK. Les encres cyan, magenta, jaune et noir sont utilisées pour représenter les couleurs des objets qui ne produisent pas leur propre lumière. Plus une couleur est utilisée en quantité, plus elle devient foncée dans le résultat final.[3]

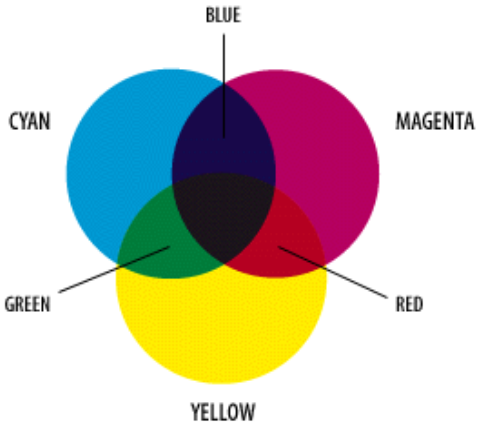

FIG. 1.6 : Le modèle de couleur CMYK

1.5.3 Gamut

Le gamut représente la totalité des couleurs qu'un appareil peut afficher ou reproduire, et il est généralement défini dans un espace colorimétrique spécifique, tel que le modèle RGB ou le modèle CMYK. Chaque modèle de couleur a ses propres limites ou plage de gamut

[2] *Cyan Magenta Yellow and Key*

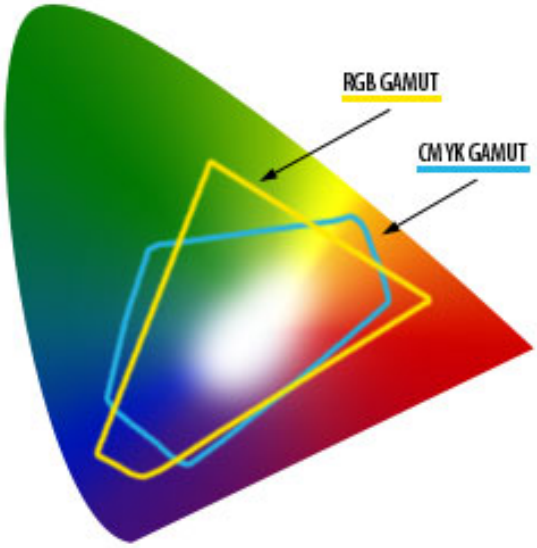

FIG. 1.7 : Gamut

1.6 Types d'image

Il existe de nombreux types d'images dans le monde. Certains des formats de fichiers les plus courants sont :

- **GIF**[3] **-** un format bitmap 8 bits (256 couleurs) compressé de manière non destructive. Principalement utilisé pour le Web. A plusieurs sous-normes dont l'une est l'animation GIF.
- **JPEG**[4] **-** un très efficace (c'est-à-dire beaucoup d'informations par octet) de manière destructive format bitmap compressé 24 bits (16 millions de couleurs). Largement utilisé, en particulier pour le web et Internet (bande passante limitée).
- **TIFF**[5] **-** le format bitmap de publication 24 bits standard. Compresse de manière non destructive avec, par exemple, la compression Lempel-Ziv-Welch (LZW).
- **PS**[6] **-** Postscript, un format vectoriel standard. A de nombreuses sous-normes et peut être difficile à transporter entre les plates-formes et les systèmes d'exploitation.
- **PSD**[7] **-** un format Photoshop dédié qui conserve toutes les informations dans un image incluant toutes les couches.

[3] *Graphics Interchange Format*
[4] *Joint Photographic Experts Group*
[5] *Tag Image File Format*
[6] *Photoshop*
[7] *Photoshop Document*

1.7 Amélioration de l'image avec des transformations de niveaux de gris

Le niveau de gris d'un pixel se réfère à l'intensité lumineuse mesurée en un point donné. Au lieu de représenter la couleur complète d'un pixel, les images en niveaux de gris utilisent une gamme de valeurs allant du noir au blanc, avec des niveaux intermédiaires définis par un nombre fini d'options.

Ainsi, pour représenter une image en niveaux de gris, chaque pixel se voit attribuer une valeur correspondant à la quantité de lumière réfléchie. Cette valeur est généralement exprimée dans une plage de 0 à 255, par exemple. Contrairement à une représentation binaire où chaque pixel est décrit par un seul bit, dans une image en niveaux de gris, chaque pixel est représenté par un octet (8 bits).

Pour pouvoir afficher correctement les différentes nuances de gris, le matériel utilisé pour l'affichage de l'image doit être capable de reproduire ces variations. Le nombre de niveaux de gris disponibles dépend du nombre de bits utilisés pour décrire la "couleur" de chaque pixel de l'image. En augmentant le nombre de bits, on augmente également le nombre de niveaux de gris possibles, permettant ainsi une représentation plus précise et détaillée des nuances de luminosité.

1.8 Seuillage d'image

Le seuillage est une opération qui convertit les pixels d'une image en valeurs binaires en fonction d'un seuil prédéfini. Les pixels en dessous du seuil deviennent noirs, tandis que ceux au-dessus deviennent blancs. Cela permet de mettre en évidence des formes ou des objets dans l'image. Cependant, le choix du seuil approprié est crucial et peut être complexe, car il dépend de divers facteurs tels que le contenu de l'image et l'éclairage. Il est nécessaire d'expérimenter avec différents seuils pour obtenir des résultats optimaux.[4]

- **Seuillage globale :** La méthode de segmentation par seuil global implique l'utilisation d'un seuil unique pour l'ensemble de l'image, en attribuant la même valeur de gris. En général, le seuillage est une technique qui vise à déterminer de manière optimale la valeur du seuil en fonction d'un critère spécifique. [4]

- **Seuillage locale :** L'objectif de seuillage dans cette méthode est de définir un seuil pour chaque pixel en se basant sur la luminosité des pixels voisins. Les approches de segmentation par seuillage local impliquent de diviser l'image en blocs de taille égale, généralement de dimensions 32x32. Ensuite, un seuil optimal est calculé pour chaque bloc en utilisant l'une des méthodes de seuillage globales.[4]

1.9 Lissage des images

Les filtres passe-bas sont des outils utilisés pour atténuer les hautes fréquences d'un signal, réduisant ainsi le bruit provenant de différentes sources. Ils sont particulièrement

importants dans la détection de contours. Techniquement, ce sont des approximations discrètes de filtres continus. Ils ne modifient pas le niveau global du signal, et les valeurs des entrées de la matrice de convolution sont généralement des nombres entiers divisés par leur somme. Cela permet d'obtenir une version filtrée du signal, préservant les caractéristiques essentielles tout en réduisant les composantes indésirables. [5]

- **Filtre uniforme :** Ce filtre est créé en effectuant une convolution entre deux filtres unidimensionnels rectangulaires. Chaque élément de la matrice a la même valeur. Cependant, l'inconvénient de ce filtre est qu'il provoque un déphasage.
- **Filtre pyramidal :** Lorsque vous combinez un filtre rectangulaire avec lui-même, cela se traduit par un filtre triangulaire qui maintient les phases constantes. Le filtre pyramidal est créé à partir des filtres triangulaires dans les deux directions.
- **Filtre gaussien :** ce filtre très populaire utilise la loi de probabilité de Gauss, également connue sous le nom de loi normale multidimensionnelle, est utilisée dans ce filtre largement répandu. En utilisant le théorème central limite, il est possible d'itérer l'un des filtres précédents pour obtenir des approximations de plus en plus précises.

1.10 Contours dans une image

Les filtres utilisés dans ce processus transforment l'image d'origine en une version où la plupart des pixels sont noirs, sauf aux endroits où un contour est identifié, ce qui est représenté par des pixels blancs. Les valeurs exactes des pixels ne sont pas significatives, et il n'est pas nécessaire de les mettre à l'échelle, contrairement à ce qui peut être fait lors d'un processus de lissage.[6]

La détection repose sur le calcul des dérivées par rapport aux deux coordonnées. Lorsqu'on considère les signaux comme des combinaisons de sinusoïdes, la dérivation agit comme un filtre passe-haut, ce qui peut introduire du bruit et créer de faux contours. Pour minimiser ce problème, il est conseillé aux amateurs d'atténuer le bruit en appliquant d'abord un filtre flou avant d'utiliser un filtre simple. Les professionnels ont développé des méthodes plus avancées et systématiques pour résoudre ce problème. [6]

- **Filtre dérivées premières :** Une méthode basique de filtrage consiste à calculer les variations entre les pixels adjacents horizontalement et verticalement. Chaque point extrême correspond alors à un point constituant un contour.
- **Filtre de Prewitt :** Le filtre de Prewitt est composé de deux matrices qui combinent un filtre de dérivation dans une direction donnée avec un filtre de flou rectangulaire dans l'autre direction. Cette combinaison introduit un effet de flou dans le traitement de l'image.
- **Filtre de Sobel :** Une amélioration de la technique consiste à substituer le filtre rectangulaire par un filtre triangulaire.

- **Filtre de Canny :** Le filtre de Sobel, qui est précédé d'un lissage gaussien et suivi d'un seuillage, est conçu pour atteindre l'optimalité selon trois critères importants.
- **Filtre de Deriche :** Une alternative efficace au filtre de Canny, offrant des résultats comparables.
- **Filtre dérivées secondes :** Les différences finies sont couramment utilisées pour calculer ces valeurs, où un changement de signe correspond à un point caractéristique du contour. Leur somme, connue sous le nom de laplacien, est souvent utilisée pour exploiter ces informations.
- **Filtre de Marr-Hildreth :** Avant de calculer le laplacien, il est courant de prétraiter les données en utilisant un lissage gaussien. Ce lissage utilise deux variances ajustables afin de filtrer les hautes fréquences présentes dans les données.

1.11 Histogrammes d'une image

L'histogramme des niveaux de gris ou des couleurs d'une image est une fonction qui donne la fréquence d'apparition de chaque niveau de gris (couleur) dans l'image. Il permet de donner un grand nombre d'information sur la distribution des niveaux de gris (couleur) et de voir entre quelles bornes est repartie la majorité des niveaux de gris (couleur) dans le cas d'une image trop claire ou d'une image trop foncée. Donc, c'est la fonction discrète "h" telle que "h(i) = n", où n est le nombre de pixels de l'image ayant l'intensité i, que l'on appelle « effectif ».

Il peut être utilisé pour améliorer la qualité de l'image (augmentation d'image) en introduisant quelques modifications afin de pouvoir en extraire des informations utiles. Pour réduire les erreurs de quantification, comparer deux images acquises sous des éclairages différents, ou mesurer certaines propriétés des images.

1.12 Segmentation des images

La segmentation d'image est une méthode de division d'une image numérique en sous-groupes appelés segments d'image, réduisant la complexité de l'image et permettant un traitement ou une analyse plus poussé de chaque segment d'image. Techniquement, la segmentation est l'attribution d'étiquettes aux pixels pour identifier des objets, des personnes ou d'autres éléments importants de l'image.

Une utilisation courante de la segmentation d'image est dans la détection d'objets. Au lieu de traiter l'image entière, une pratique courante consiste à utiliser d'abord un algorithme de segmentation d'image pour trouver des objets d'intérêt dans l'image. Ensuite, le détecteur d'objet peut opérer sur une boîte englobante déjà définie par l'algorithme de segmentation. Cela empêche le détecteur de traiter l'intégralité de l'image, ce qui améliore la précision et réduit le temps de référence.

La segmentation d'images est un élément clé des technologies et des algorithmes de vision par ordinateur. Il est utilisé pour de nombreuses applications pratiques, notam-

ment l'analyse d'images médicales, la vision par ordinateur pour les véhicules autonomes, la reconnaissance et la détection des visages, la vidéo-surveillance et l'analyse d'images satellite.

Il existe différentes manières de segmenter une image. Voici quelques-unes des principales techniques :

- **Segmentation sémantique :** La segmentation sémantique vise à regrouper les pixels d'une image en fonction de leurs classes sémantiques. Dans ce type de modèle, chaque pixel est attribué à une classe spécifique, sans considérer d'autres informations ou contextes. Par exemple, lorsqu'on applique la segmentation sémantique à une image comportant des arbres et des véhicules, on obtient un masque qui regroupe tous les types d'arbres en une seule classe (par exemple, "arbre") et tous les types de véhicules (bus, voitures, vélos) en une seule classe (par exemple, "véhicules").

  Cette approche présente cependant certains inconvénients, notamment en ce qui concerne la définition précise du problème, surtout lorsqu'il existe plusieurs instances regroupées dans une même classe. Par exemple, une image d'une rue bondée pourrait segmenter toute la zone où se trouve la foule en une seule classe appelée "personnes". La segmentation sémantique ne fournit pas de détails approfondis dans des images complexes de ce type.

  En d'autres termes, bien que la segmentation sémantique permette de catégoriser les pixels en fonction de classes sémantiques générales, elle peut être limitée pour fournir une granularité plus fine dans des scènes complexes où plusieurs objets de la même classe se trouvent dans une région donnée.

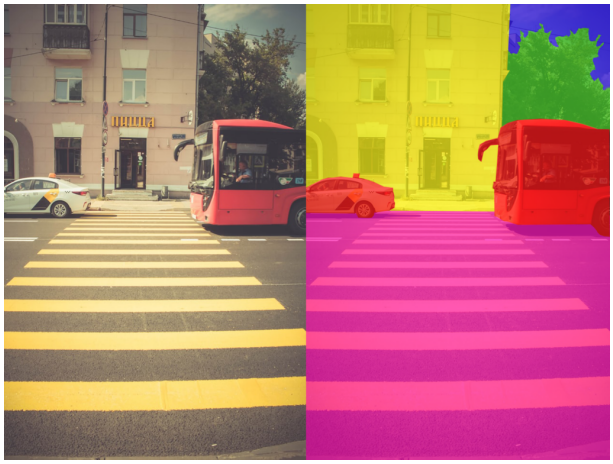

Fig. 1.8 : Segmentation sémantique.

- **Segmentation des objets :** La segmentation des objets vise à regrouper les pixels en fonction des objets présents dans une image, sans pour autant attribuer de classe spécifique à chaque région. Les algorithmes de segmentation d'instance ne sont pas conscients de l'appartenance de chaque région à une classe particulière. Au lieu de cela, ils cherchent à séparer les régions similaires ou qui se chevauchent en se basant sur les contours des objets.

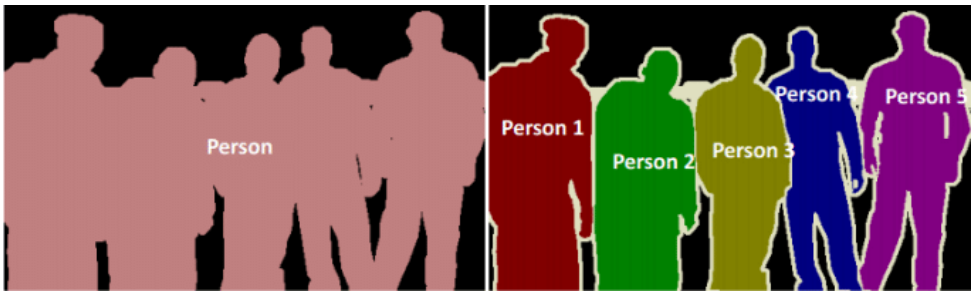

FIG. 1.9 : Segmentation sémantique (gauche) et segmentation d'instance (droite)

- **segmentation panoptique :** La segmentation panoptique est une approche récente de segmentation qui combine la segmentation sémantique et la segmentation d'instance. Son objectif est de prédire l'identité de chaque objet et de séparer chaque instance de ces objets dans une image donnée.

  La segmentation panoptique joue un rôle crucial dans de nombreux domaines où une grande quantité d'informations est nécessaire pour accomplir des tâches spécifiques. Par exemple, les véhicules autonomes doivent être en mesure de percevoir et de comprendre rapidement et précisément leur environnement. Pour cela, ils peuvent utiliser un algorithme de segmentation panoptique pour analyser en temps réel un flux d'images.

  Cette technique permet aux voitures autonomes de reconnaître et de catégoriser différents objets présents sur la route, tels que les piétons, les voitures, les panneaux de signalisation, les feux de circulation, etc. En identifiant chaque instance d'objet, la segmentation panoptique fournit des informations détaillées sur la localisation et les caractéristiques spécifiques de chaque élément dans la scène.

1.13 Conclusion

La représentation des images numériques est l'un des éléments fondamentaux des applications multimédia. Cependant, le traitement d'images pose des problèmes beaucoup plus complexes que le traitement de texte. En fait, une image est un objet à deux dimensions qui doit représenter un espace à trois dimensions, qui a deux conséquences principales, puisque la quantité de données à traiter est beaucoup plus importante, et la structure de ces données est beaucoup plus complexe. Par conséquent, le traitement, le stockage et la représentation de ces données se heurte à certaines limites. Ces limitations sont contournées grâce au traitement d'image.

Chapitre 2

Extraction de caractéristiques pour la reconnaissance des objets

2.1 Introduction

L'extraction des caractéristiques et la reconnaissance d'objets joue un rôle essentiel dans le domaine du traitement d'images et de la vision par ordinateur. Ces caractéristiques permettent de représenter de manière concise et distinctive les objets présents dans une image, facilitant ainsi leur identification et leur reconnaissance. Différentes méthodes de détection d'objets sont utilisées, telles que la détection de mouvement, la détection de contours, la segmentation d'images, et bien d'autres. Ces méthodes permettent de mettre en évidence les propriétés et les attributs des objets, facilitant ainsi leur analyse et leur compréhension dans le contexte de l'image traitée. Le choix de la méthode appropriée dépendra des caractéristiques spécifiques aux des objectifs de reconnaissance des objets.

2.2 Extraction de caractéristiques

L'extraction de caractéristiques est une étape clée dans le traitement d'images et la reconnaissance d'objets. Cette étape consiste à extraire des informations significatives et spécifiques des images afin de les utiliser pour des tâches de détection, de classification ou d'analyse.

Il existe différentes caractéristiques qui peuvent être exploitées en fonction des besoins spécifiques de l'application. Parmi les caractéristiques couramment utilisées, on peut citer :

- **Descripteurs de forme :** Ces caractéristiques permettent de représenter la forme d'un objet en utilisant des mesures géométriques telles que l'aire, le périmètre, les moments de forme, etc.
- **Descripteurs de texture :** Ces caractéristiques captent les variations de texture dans une image, en utilisant des mesures statistiques ou des méthodes basées sur des filtres de texture.
- **Descripteurs de couleur :** Ces caractéristiques représentent les informations de couleur dans une image, notamment à travers des histogrammes de couleurs, des moments de couleur ou des descripteurs basés sur les espaces colorimétriques.
- **Descripteurs basés sur les contours :** Ces caractéristiques exploitent les propriétés des contours dans une image, en utilisant des mesures telles que la longueur des contours, les orientations, les dérivées de contours, etc.
- **Descripteurs basés sur l'apprentissage automatique :** Ces caractéristiques sont apprises à partir des données d'entraînement à l'aide d'algorithmes d'apprentissage automatique tels que les réseaux de neurones convolutionnels (CNN) ou les descripteurs de type bag-of-features.

Il est important de choisir les caractéristiques appropriées en fonction de la tâche de reconnaissance d'objets spécifique et des caractéristiques des images traitées. La combinaison de différentes caractéristiques peut également être utilisée pour améliorer les performances de la reconnaissance d'objets.

2.2.1 C'est quoi les caractéristiques dans une image ?

L'image est une composition organisée de données définie par les paramètres suivants :

a) Pixel :

Le terme "pixel" est une contraction de l'expression anglaise "Picture Element", ce qui signifie élément d'image. Il représente le plus petit point constituant une image. Un pixel est une entité pouvant être calculée, qui peut être structurée et quantifiée. Si un bit est la plus petite unité d'information qu'un ordinateur peut traiter, le pixel est le plus petit élément manipulable par les dispositifs matériels et logiciels d'affichage ou d'impression. Par exemple, la lettre A peut être représentée par un groupe de pixels, comme illustré dans la figure ci-dessous :

FIG. 2.1 : La lettre A affichée comme un groupe de pixels.[7]

b) Dimension :

La taille d'une image se réfère à ses dimensions. Elle est représentée sous forme d'une matrice où chaque élément correspond à une valeur numérique représentative de l'intensité lumineuse d'un pixel. En multipliant le nombre de lignes de cette matrice par le nombre de colonnes, on obtient le nombre total de pixels présents dans l'image.

c) Résolution :

La résolution d'une image est déterminée par le nombre de pixels par unité de longueur, mesuré en dpi (dots per inch) ou ppp (points par pouce). On utilise le terme "définition" pour un écran et "résolution" pour une image. Lorsque le nombre de pixels par unité de longueur est élevé, cela signifie qu'il y a plus d'informations pour décrire l'image, ce qui se traduit par une résolution plus élevée (et un poids d'image plus élevé).[7]

La résolution d'une image indique le niveau de détail qui sera représenté sur cette image. Lors de la numérisation, deux équations sont prises en compte :

(X * résolution) = x pixels , **(Y * résolution)** = y pixels

Dans ces équations :

- X et Y représentent la taille de la structure à numériser en pouces ou en centimètres (1 pouce = 2,54 centimètres).
- La résolution représente la résolution de numérisation.
- x et y représentent la taille de l'image en pixels.

d) La taille de l'image :

Afin de déterminer la dimension d'une image, il est nécessaire de compter le nombre de pixels qu'elle contient. Cela équivaut à calculer la multiplication entre la hauteur et la largeur de l'image, ce qui représente le nombre de cases dans une grille. La taille totale de l'image est alors obtenue en multipliant ce nombre de pixels par la taille en octets de chaque pixel.

Par exemple, prenons une image de dimensions **240 * 420** en mode TrueColor. Le nombre de pixels dans cette image est calculé comme suit :

240 * 420 = 100800 pixels

La taille de chaque pixel est de 24 bits, soit 3 octets :

Taille de chaque pixel : **24 bits / 8 = 3 octets**

Par conséquent, le poids total de l'image est calculé comme suit :

100 800 * 3 = 302400 octets, ce qui équivaut à **302400/1024 = 295 Ko**.

e) Bruit :

Le bruit dans une image est généralement causé par des fluctuations soudaines de l'intensité d'un pixel par rapport à ses voisins. Il est souvent dû à des variations dans l'éclairage provenant des dispositifs optiques et électroniques du capteur.

f) Contraste :

Le contraste est une mesure qui met en évidence la différence notable entre deux zones distinctes d'une image, en particulier entre les zones sombres et les zones claires de celle-ci. Il est évalué en se basant sur les niveaux de luminance de ces deux zones de l'image.

g) Luminance :

La luminance fait référence à la quantité de lumière présente dans les points d'une image. Elle peut être définie comme le rapport entre l'intensité lumineuse d'une surface et son aire apparente, pour un observateur éloigné. Dans ce contexte, le terme "luminance" est utilisé pour décrire la luminosité d'une surface, tandis que "brillance" se réfère à l'éclat d'un objet.

Une bonne luminance se caractérise par plusieurs aspects importants :

- **Des images lumineuses et brillantes :** cela signifie que les points de l'image ont une intensité lumineuse adéquate, ce qui crée une impression de luminosité et de clarté.
- **Un bon contraste :** il est préférable d'éviter les images présentant une gamme de contraste excessive qui tend vers le blanc ou le noir. Ces types d'images peuvent entraîner une perte de détails dans les zones sombres ou lumineuses, ce qui peut nuire à la qualité globale de l'image.
- **L'absence de parasites :** une bonne luminance implique également l'absence de distorsions ou d'artefacts visuels indésirables, tels que des parasites, du bruit ou des fluctuations de luminosité incohérentes.

2.2.2 Extraction des caractéristiques

Les caractéristiques d'une image sont des éléments qui permettent de décrire de manière précise une image, et leur importance dépend de l'analyse caractéristique effectuée. Ces points d'intérêt peuvent inclure des points d'intérêt, des régions d'intérêt, des coins, des lignes, la couleur, les bords, les blobs, la texture, etc. Ces différentes caractéristiques présentent l'avantage d'avoir une signification sémantique riche en informations et de rester stables malgré les transformations géométriques.

Dans cette étude, les caractéristiques utilisées se concentrent sur la texture et les points d'intérêt, en exploitant leur spécificité d'analyse.

a) Points d'intérêt

Dans une image, on peut repérer des points d'intérêt qui correspondent à des endroits où la fonction d'intensité présente des changements brusques et distincts. Ces changements peuvent se manifester sous la forme de coins, de régions, de jonctions ou de zones avec une texture fortement variée. Ces points d'intérêt, qui possèdent des caractéristiques spécifiques, sont fréquemment utilisés dans la reconnaissance faciale.[8]

La détection de points d'intérêt joue un rôle crucial en tant qu'étape préliminaire dans de nombreux processus de reconnaissance utilisés en vision par ordinateur. Les points d'intérêt extraits sont fréquemment utilisés dans des tâches telles que la reconstruction d'images, le suivi d'objets, la recherche d'images, la reconnaissance de mouvements, la reconnaissance faciale, et bien d'autres encore. Ces points d'intérêt permettent de localiser et de représenter de manière significative des caractéristiques clés dans les données visuelles, facilitant ainsi leur traitement et leur utilisation ultérieure dans différentes applications.

Les points d'intérêt [9]jouent un rôle essentiel dans de nombreuses applications de reconnaissance, notamment la biométrie, le contrôle qualité, les systèmes d'informations géographiques (SIG), la médecine, la robotique, et bien d'autres encore. Ils sont largement utilisés en raison de leur utilité et de leur efficacité dans ces domaines variés.

Les points d'intérêt dans une image se réfèrent à des zones où il y a des variations brusques dans la fonction d'intensité. Ces variations peuvent être causées par des changements de réflectance ou de profondeur. L'utilisation de points d'intérêt présente plusieurs

avantages. Tout d'abord, ces points fournissent une source d'informations fiables en se basant sur la fonction d'intensité. De plus, ils sont considérés comme robustes car ils sont invariants aux transformations et sont clairement localisés. Enfin, de nombreux points d'intérêt sont présents dans de nombreuses images.

b) Texture

La reconnaissance visuelle des textures peut sembler facile, mais sa définition demeure complexe. Dans la littérature, plusieurs définitions sont proposées en fonction du contexte d'étude.[10]

La texture peut être définie de différentes manières selon le domaine d'application et les auteurs. En termes simples, la texture se réfère à une région plus large qui présente une structure composée de motifs répétitifs. Ces motifs sont formés par des éléments de base appelés "texels" ou primitives, qui sont organisés selon une règle de placement spécifique.[11]

La définition de la texture est un sujet qui suscite diverses interprétations et il existe plusieurs définitions dans la littérature. Cette pluralité de définitions s'explique par le fait que chaque définition prend en compte des critères spécifiques. Les textures sont des motifs visuels complexes composés d'entités ou de petits motifs qui possèdent des caractéristiques telles que la luminosité, la couleur, la pente, la granularité, la taille, et bien d'autres encore. Ainsi, la texture peut être considérée comme un regroupement d'éléments similaires de petite taille dans une image. Les propriétés locales de ces petits motifs peuvent être la légèreté, l'uniformité, la densité, la rugosité, la régularité, la linéarité, la fréquence, la phase, l'orientation, la grossièreté, le caractère aléatoire, la finesse, la granularité, et ainsi de suite. La figure présente des exemples de textures.

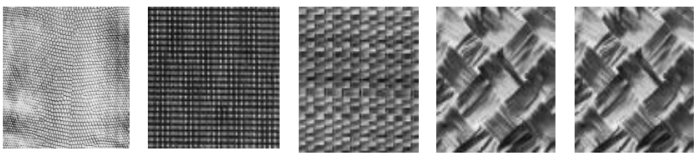

Fig. 2.2 : Quelques exemples de textures.[12]

2.3 Détection et reconnaissance des objets

2.3.1 Qu'est-ce que la reconnaissance d'objet ?

La reconnaissance d'objets est une méthode de vision par ordinateur qui vise à identifier les objets présents dans des images et des vidéos. Elle repose sur l'utilisation d'algorithmes de Deep Learning et d'apprentissage automatique. Lorsque les humains observent des photos ou regardent des vidéos, ils sont capables de reconnaître instantanément les personnes, les objets, les scènes et les détails visuels présents. L'objectif est d'enseigner à un ordinateur à effectuer cette tâche naturelle pour les humains et à lui permettre de comprendre adéquatement le contenu de l'image. [13]

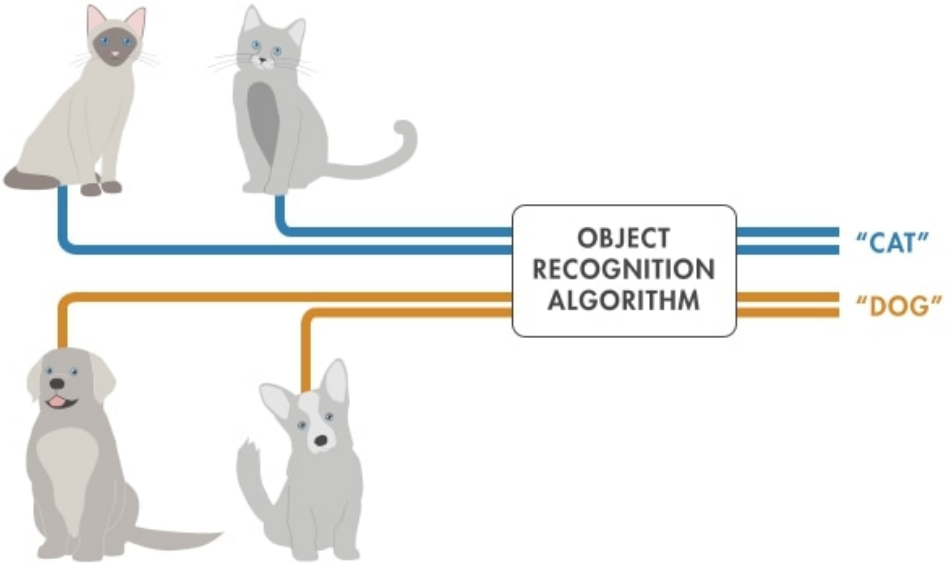

Fig. 2.3 : Système de la reconnaissance d'objets. [13]

La reconnaissance d'objets est une discipline de la vision artificielle et joue un rôle essentiel dans la vision par ordinateur. Son objectif est d'identifier des formes préalablement définies dans une image numérique, ainsi que dans un flux vidéo numérique.

Il est important de distinguer la reconnaissance d'objets (ou "object recognition" en anglais) de la reconnaissance de formes ("pattern recognition" en anglais). La première consiste à reconnaître des formes géométriques dans une image, tandis que la seconde vise à identifier des motifs dans des données statistiques. Il arrive souvent que la reconnaissance de formes soit utilisée comme une technique appliquée à la reconnaissance d'objets, ce qui peut causer une certaine confusion.[13]

Initialement, la reconnaissance d'objets était basée sur des algorithmes conçus par des humains, jusqu'en 1995, avec des tentatives de reproduire le raisonnement humain d'identification, par exemple en disant qu'un vélo possède deux roues et un cadre. Cependant, depuis lors, d'importants progrès ont été réalisés grâce à l'utilisation de techniques d'apprentissage, telles que les séparateurs à vaste marge. Ces techniques exploitent des bases d'exemples positifs et négatifs (contre-exemples) pour permettre à un algorithme de rechercher des critères discriminants, c'est-à-dire des critères qui permettent de séparer au mieux les exemples des contre-exemples.[13]

2.4 Classification des images

La classification d'images est une branche de l'intéligence artificielle couramment utilisée pour analyser des données visuelles, dans laquelle des algorithmes d'apprentissage automatique sont employés afin d'assigner des étiquettes ou des catégories à des images. Cette méthode trouve une large gamme d'applications, allant de la reconnaissance d'objets dans des images à la détection de maladies à partir d'images médicales.

Les algorithmes de classification d'images reposent généralement sur des modèles d'ap-

prentissage automatique tels que les réseaux de neurones convolutifs (CNN[1]) et le modèle YOLO[2]. Ces modèles sont entraînés à reconnaître des motifs dans les images en utilisant de vastes ensembles de données d'entraînement étiquetées. Une fois entraînés, les modèles peuvent être utilisés pour classifier de nouvelles images en fonction des étiquettes apprises pendant la phase d'entraînement.

La classification d'images peut être réalisée selon différentes méthodes, comprenant la classification binaire (par exemple, déterminer si une image contient ou non un objet spécifique), la classification multiclasse (par exemple, attribuer une étiquette à une image parmi plusieurs classes possibles).

2.5 Localisation d'objet

La localisation d'objets est une méthode d'analyse d'images visant à détecter la position et la forme d'un ou plusieurs objets dans une image. Cette technique est largement utilisée dans divers domaines tels que la surveillance de sécurité, la détection de visages, la détection de panneaux de signalisation routière, et bien d'autres.

La localisation d'objets est souvent réalisée à l'aide de réseaux de neurones convolutifs (CNN), qui sont des modèles d'apprentissage profond capables d'extraire des caractéristiques essentielles des images. Les CNN peuvent être entraînés sur de grandes quantités de données étiquetées afin d'identifier les caractéristiques clés des objets à détecter. Une fois entraînés, ces modèles peuvent être utilisés pour détecter la présence et la position des objets dans de nouvelles images.

Plusieurs approches sont utilisées pour la localisation d'objets, notamment la détection de boîtes englobantes, la segmentation sémantique et la détection d'objets en temps réel. La détection de boîtes englobantes consiste à dessiner des rectangles autour des objets détectés pour les encadrer. La segmentation sémantique permet de diviser l'image en plusieurs parties et d'attribuer une étiquette sémantique à chaque partie. La détection d'objets en temps réel permet de détecter et de suivre les objets en temps réel dans une séquence vidéo.

Ces techniques de localisation d'objets sont continuellement améliorées grâce aux avancées de l'intelligence artificielle et de l'apprentissage automatique. Elles jouent un rôle crucial dans de nombreux systèmes et applications qui nécessitent une compréhension précise de l'environnement visuel, ouvrant ainsi la voie à des développements passionnants dans des domaines tels que la robotique, la réalité augmentée, la conduite autonome et bien d'autres encore.

2.6 Détection d'objet

La détection d'objets est une méthode de traitement d'images et de vidéos qui permet d'identifier et de localiser des éléments spécifiques au sein de ces médias. Les algorithmes

[1] *Convolutional Neural Network*
[2] *You Only Look Once*

de détection d'objets exploitent généralement l'apprentissage automatique ou l'apprentissage profond pour obtenir des résultats significatifs. Lorsque les êtres humains observent des images ou des vidéos, ils sont capables de reconnaître et de localiser des objets d'intérêt en un clin d'œil. L'objectif de la détection d'objets est de reproduire cette capacité cognitive à l'aide d'un ordinateur.

2.6.1 Pourquoi la détection d'objets est importante ?

La détection d'objets est une technologie essentielle utilisée dans les systèmes avancés d'aide à la conduite (ADAS[3]), la vidéosurveillance et les systèmes de récupération d'images. Elle permet aux véhicules d'améliorer la sécurité routière en détectant les voies de circulation et les piétons, tout en permettant la surveillance et la recherche efficace d'objets dans les images. Les progrès de la vision par ordinateur et de l'apprentissage automatique ont permis des avancées significatives dans ce domaine.

FIG. 2.4 : Utilisation de la détection d'objets pour identifier et localiser les véhicules.

La détection d'objets est importante pour de nombreuses applications, notamment :

- **Surveillance et sécurité :** Dans le domaine de la surveillance et de la sécurité, la détection d'objets joue un rôle essentiel en permettant de repérer les intrusions, les objets suspects ou les comportements anormaux. Les systèmes de surveillance équipés de modèles de détection d'objets sont capables de signaler toute situation suspecte aux agents de sécurité, ce qui facilite une intervention rapide et efficace.
- **Automobile et transport :** La détection d'objets revêt une grande importance dans le domaine de la conduite autonome et des transports. Les systèmes de détection d'objets permettent d'identifier les piétons, les véhicules, les motos, les panneaux de signalisation, les feux de circulation et autres obstacles présents sur la route, contribuant ainsi à une conduite plus sécurisée.
- **Médecine et recherche :** Dans le domaine médical, la détection d'objets est largement utilisée pour l'analyse d'images médicales et la détection de tumeurs, d'anomalies et d'autres éléments d'intérêt clinique. Elle est également employée dans la recherche scientifique, notamment pour détecter des objets dans les images capturées par des télescopes dans le domaine de l'astronomie.

[3] *Advanced Driver Assistance Systems*

- **Fabrication et logistique :** Dans les processus de fabrication et de logistique, la détection d'objets est utilisée pour repérer les produits défectueux, détecter les objets manquants et assurer le suivi des produits tout au long de la chaîne d'approvisionnement.
- **Commerce électronique et marketing :** La détection d'objets trouve également des applications dans le domaine du commerce électronique et du marketing en ligne. Elle permet de reconnaître les produits présents dans des images, ce qui facilite la recherche de produits et améliore l'expérience utilisateur lors des achats en ligne.

2.7 Modelés de reconnaissance d'objet

2.7.1 Régions avec modèle CNN

Un réseau neuronal convolutif (CNN ou ConvNet) est une architecture réseau pour le Deep Learning qui apprend directement à partir des données.

Ce type de réseau est particulièrement efficace lorsqu'il s'agit de trouver des patterns dans des images afin de reconnaître des objets, des classes et des catégories. Les CNN peuvent aussi être efficaces pour la classification de données audio, de séries temporelles et de signaux.

L'algorithme R-CNN d'origine se décompose en quatre étapes distinctes dans son processus.[14]

Voici une reformulation des étapes de l'algorithme R-CNN d'origine :

- **Dans la première étape :** une image est introduite dans le réseau.
- **La deuxième étape :** consiste à extraire des propositions de régions à partir de l'image. Ces régions sont des zones potentielles pouvant contenir des objets d'intérêt. Cette étape peut être réalisée à l'aide d'un algorithme tel que la Recherche Sélective (Selective Search).
- **À la troisième étape :** l'apprentissage par transfert est utilisé pour extraire les caractéristiques de chaque proposition de région à l'aide d'un réseau de neurones convolutif (CNN). Cette extraction de caractéristiques permet de représenter les régions de manière appropriée pour la classification ultérieure.
- **À la quatrième étape :** chaque proposition de région est classifiée en utilisant les caractéristiques extraites, en utilisant un modèle de machine à vecteur de support (SVM[4]) qui assigne des étiquettes prédites à chaque région.

Toutefois, la méthode R-CNN présente un inconvénient majeur : sa lenteur. De plus, cette approche ne permet pas véritablement d'apprendre à localiser des objets à l'aide d'un réseau de neurones profond. Au lieu de cela, elle repose sur un système plus avancé qui combine le descripteur linéaire HOG (Histogram of Oriented Gradients) avec un classificateur SVM (Support Vector Machine).[15]

[4] *Support Vector Machines*

2.7.2 Fast R-CNN :

Une amélioration de l'algorithme R-CNN d'origine a été proposée sous la forme de l'algorithme Fast R-CNN qui a été rendu public.[16]

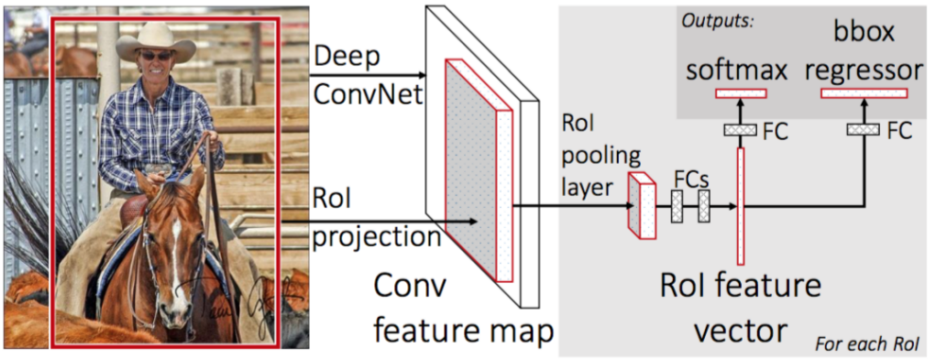

FIG. 2.5 : L'architecture Fast R-CNN. [17]

Le ROI Pooling extrait une fenêtre de taille fixe à partir d'une "Features Map" et utilise ces caractéristiques pour obtenir à la fois l'étiquette de classe finale et le cadre de délimitation. L'avantage principal de cette approche est que le réseau peut maintenant être entraîné de manière end-to-end, c'est-à-dire en intégrant toutes les étapes du processus.

- Saisir une image ainsi que les boîtes de délimitation de la vérité terrain correspondantes.
- Obtenir la carte des caractéristiques.
- Effectuer le pooling de région d'intérêt (ROI) pour générer un vecteur de caractéristiques ROI.
- Appliquer deux ensembles de couches entièrement connectées pour générer à la fois les prédictions des étiquettes de classe et les coordonnées des cadres de délimitation pour chaque proposition.

2.7.3 Faster R-CNN :

Dans l'ensemble, l'architecture Faster R-CNN permet d'atteindre une vitesse de traitement d'environ 7 à 10 images par seconde (FPS), ce qui représente une avancée considérable pour la détection d'objets en temps réel grâce à l'apprentissage profond.

2.7.4 Mask R-CNN :

Le masque R-CNN étend le populaire Faster R-CNN en apportant deux contributions majeures qui sont les suivantes : [14]

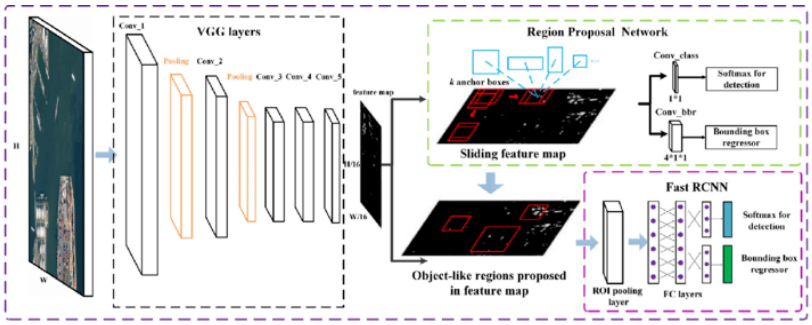

FIG. 2.6 : L'architecture Faster R-CNN. [18]

- Au lieu d'utiliser le module ROI Pooling, le module ROI Align plus précis est utilisé, ce qui permet une localisation plus précise des caractéristiques à l'intérieur des régions d'intérêt (ROI[5]).
- En plus de cela, une branche supplémentaire est insérée en dehors du module ROI Align.

Une architecture d'apprentissage en profondeur appelée Mask R-CNN a été développée par les chercheurs. Elle permet de générer un masque pixel par pixel pour chaque objet présent dans une image. Pour ce faire, une branche supplémentaire est ajoutée à l'architecture qui prend en entrée la sortie de l'étape de ROI Align, puis la fait passer à travers deux couches de convolution (CONV). Finalement, la sortie des couches de convolution correspond au masque proprement dit.

Comparaison entre les réseaux de neurones :

Voici un tableau de comparaison entre les réseaux de neurones :

[5] *Region Of Interest*

Model	Caractéristiques	Temps / image de prédiction	Limites
CNN	L'image est subdivisée en plusieurs régions, puis chaque région est classifiée dans des catégories distinctes.	-	Pour obtenir des prédictions précises, il est nécessaire d'utiliser un grand nombre de régions, ce qui entraîne un temps de calcul élevé.
R-CNN	Utiliser l'algorithme de recherche sélective pour générer des régions candidates, à partir desquelles environ 2000 régions sont extraites pour chaque image.	40-50 secondes	En raison du traitement individuel de chaque région par le CNN et de l'utilisation de trois modèles distincts pour les prédictions, le temps de calcul est considérablement élevé.
Fast R-CNN	Une seule passe de chaque image est effectuée par le CNN pour extraire les cartes d'entités. La Recherche Sélective est ensuite appliquée sur ces cartes pour générer des prédictions. Les trois modèles utilisés dans R-CNN sont ensuite combinés ensemble.	2 secondes	En raison du calcul intensif impliqué dans le traitement séparé de chaque région par le CNN, ainsi que de l'utilisation de trois modèles distincts pour les prédictions, le temps de calcul est considérablement élevé.
Faster R-CNN	En remplaçant la méthode de recherche sélective par un réseau de propositions de régions, l'algorithme a connu une nette amélioration de sa vitesse d'exécution.	0.2 secondes	La proposition d'objet est chronophage et les performances des systèmes ultérieurs sont conditionnées par celles du système précédent.
Mask R-CNN	Le modèle s'appuie sur Faster R-CNN et ajoute la capacité de segmentation des instances par pixel, en plus de la détection des objets.	< 0.2 secondes	Pour corriger le léger décalage entre les régions de la carte d'entités sélectionnées par RoIPool et les régions de l'image d'origine, il est nécessaire de remplacer RoIPool par RoIAlign.

Tab. 2.1 : Comparaison entre les réseaux de neurones

2.7.5 Modèle YOLO

YOLO est un modèle populaire de détection d'objets et de segmentation d'images, a été développé par Joseph Redmon et Ali Farhadi de l'Université de Washington. Lancé en 2015, YOLO a rapidement gagné en popularité pour sa grande vitesse et sa précision.

YOLO : (une brève histoire)

- **YOLOv2 :** lancé en 2016, a amélioré le modèle original en incorporant la normalisation par lots (batch normalization), les boîtes d'ancrage (anchor boxes) et les clusters de dimensions.[19]
- **YOLOv3 :** sorti en 2018, a encore amélioré les performances du modèle en utilisant un réseau de base plus efficace, des ancres multiples et un pooling pyramidal spatial (spatial pyramid pooling).[19]
- **YOLOv4 :** a été publié en 2020, introduisant des innovations telles que l'augmentation de données Mosaic, une nouvelle tête de détection sans ancre (anchor-free detection) et une nouvelle fonction de perte (loss function).[19]
- **YOLOv5 :** a encore amélioré les performances du modèle et ajouté de nouvelles fonctionnalités telles que l'optimisation des hyperparamètres, le suivi expérimental intégré et l'exportation automatique vers des formats d'exportation populaires.[19]
- **YOLOv6 :** a été rendu open source par Meituan en 2022 et est utilisé dans de nombreux robots de livraison autonomes de l'entreprise.[19]
- **YOLOv7 :** a ajouté des tâches supplémentaires telles que l'estimation de la pose sur l'ensemble de données COCO[6] keypoints.[19]
- **YOLOv8 :** développé par Ultralytics, représente la dernière version du modèle YOLO. En tant que modèle de pointe de dernière génération (SOTA[7]), YOLOv8 capitalise sur les succès des versions précédentes en introduisant de nouvelles fonctionnalités et améliorations, ce qui se traduit par des performances, une flexibilité et une efficacité accrues. YOLOv8 offre un support complet pour une gamme étendue de tâches d'intelligence artificielle en vision, incluant la détection, la segmentation, l'estimation de pose, le suivi et la classification. Cette polyvalence permet aux utilisateurs de bénéficier des capacités de YOLOv8 dans diverses applications et domaines.[19]

Fonctionnement de YOLO :

Pour commencer, le modèle réduit la taille de l'image d'entrée en la divisant en une grille, par exemple, de dimension 3×3. En utilisant un certain nombre de convolutions et de pooling sur chaque grille générée, il identifie les zones d'intérêt sous forme de rectangles.

[6] *Common Objects in Context*
[7] *State-of-the-Art*

Ensuite, pour chaque zone d'intérêt, un vecteur Y (pc, bx, by, bh, bw, c1, c2...) est calculé.

- La variable "**pc**" représente la probabilité d'existence d'un objet.
- Les variables "**bx**" et "**by**" indiquent les coordonnées du centre de la boîte.
- Les variables "**bw**" et "**bh**" représentent respectivement la largeur et la hauteur de la boîte.
- La variable "**c**" correspond au nombre de classes.

Les zones d'intérêt qui présentent une probabilité faible par rapport à une valeur prédéfinie sont éliminées. [20]

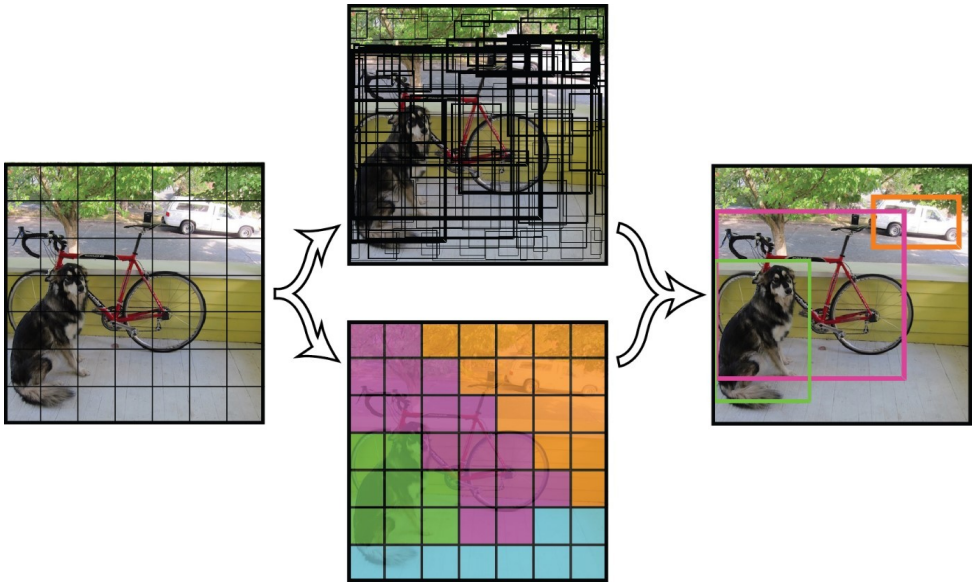

FIG. 2.7 : Fonctionnement de YOLO.

Comparaison entre les modèles YOLO :

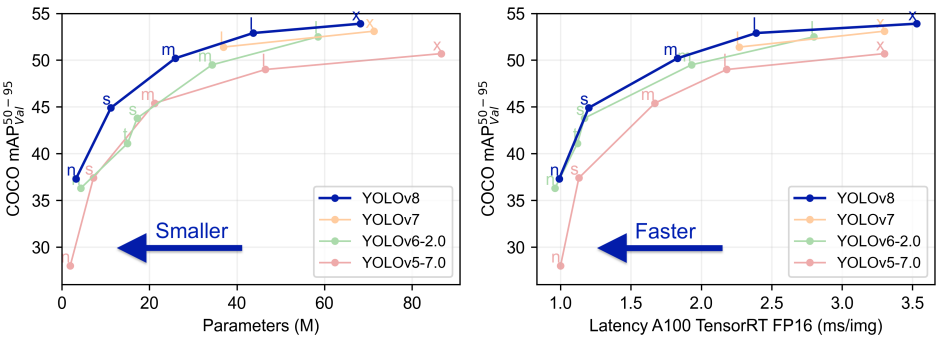

FIG. 2.8 : Comparaison entre les modèles YOLO. [19]

2.8 Conclusion

Les caractéristiques dans une image sont les informations visuelles spécifiques qui peuvent être extraites de l'image et utilisées pour la reconnaissance des objets. Cela inclut des éléments tels que les contours, les textures, les couleurs, les formes et les motifs présents dans l'image.

Cependant, avec l'avènement du Deep Learning, de nouvelles techniques d'extraction de caractéristiques ont émergé. Ces techniques utilisent des réseaux de neurones profonds, tels que les réseaux de neurones convolutifs (CNN), pour apprendre automatiquement à extraire des caractéristiques pertinentes à partir des images. Ces modèles d'extraction de caractéristiques basés sur le Deep Learning ont montré de meilleures performances et une plus grande capacité à représenter les objets de manière appropriée pour la reconnaissance.

En résumé, les caractéristiques dans une image sont les informations visuelles spécifiques utilisées pour la reconnaissance des objets. Les techniques classiques d'extraction de caractéristiques ont été améliorées grâce à l'utilisation de l'apprentissage automatique et du Deep Learning, qui permettent d'extraire automatiquement des caractéristiques pertinentes à partir des images. Ces avancées ont ouvert de nouvelles perspectives dans le domaine de la reconnaissance des objets.

Chapitre 3

Segmentation pour améliorer la reconnaissance des caractères d'une plaque d'immatriculation

3.1 Introduction

Dans ce chapitre, nous aborderons la phase de réalisation et d'implémentation de notre système après avoir examiné les méthodes d'extraction de caractéristiques pour la reconnaissance des objets. L'objectif de ce chapitre est de présenter les aspects de mise en œuvre de notre application, tels que l'environnement matériel sur lequel le système a été développé, les langages de programmation utilisés et les outils utilisés. Ensuite, nous décrirons les interfaces graphiques et les fonctionnalités de notre application, tout en fournissant un exemple concret pour illustrer les résultats obtenus avec notre approche.

3.2 Environnement matériel

Afin de mener à bien ce projet, il a été mis à notre disposition un ordinateur Dell avec les caractéristiques suivantes :

- **Processeur :** Intel(R) Core(TM) i7-7820HQ CPU @ 2.90GHz 2.90 GHz
- **RAM :** 16,00 Go DDR4
- **Disque Dur :** 512 Go SSD
- **Système d'Exploitation :** Microsoft Windows 10 Pro 64-bit.

3.3 Le langage de programmation utilisé

Dans cette section, nous allons introduire la définition du langage que nous utiliserons pour mettre en œuvre et réaliser notre système.

3.3.1 Python

Python est largement adopté dans le domaine de l'intelligence artificielle, en particulier pour le Deep Learning. Cela est dû à sa richesse en bibliothèques performantes et utiles pour des tâches telles que la vision artificielle et l'utilisation de réseaux de neurones. Python bénéficie également d'une syntaxe simple et lisible, d'une grande flexibilité et d'une vaste communauté de développeurs, ce qui en fait un choix privilégié par de nombreux chercheurs et praticiens en IA[1].

[1] *Intelligence Artificielle*

Fig. 3.1 : Logo de Python

3.4 Les bibliothèques Python

3.4.1 NumPy

NumPy est une bibliothèque Python largement utilisée pour la manipulation de tableaux. En plus de fournir des fonctionnalités pour travailler avec des tableaux, NumPy offre des outils pour l'algèbre linéaire, les transformations de Fourier et les opérations sur les matrices. NumPy a été développé par Travis Oliphant en 2005 et il s'agit d'un projet open source, ce qui signifie que vous pouvez l'utiliser gratuitement et en toute liberté.

Fig. 3.2 : Logo de NumPy

3.4.2 EasyOCR

EasyOCR est un lecteur de caractères imprimés dépendant de la police, basé sur un algorithme de correspondance de modèles. Il a été conçu pour lire différents types de courts textes tels que les numéros de pièces, les numéros de série, les dates d'expiration,

les dates de fabrication, les codes de lot, imprimés sur des étiquettes ou directement sur des pièces.

3.4.3 OpenCV

OpenCV est une bibliothèque libre, initialement développée par Intel, spécialisée dans le traitement d'images en temps réel. La société de robotique Willow Garage, puis la société ItSeez se sont succédé au support de cette bibliothèque. Depuis 2016 et le rachat de ItSeez par Intel, le support est de nouveau assuré par Intel. [21]

Fig. 3.3 : Logo de OpenCV

3.5 L'environnement de développement

3.5.1 PyCharm

PyCharm est un environnement de développement intégré (IDE) utilisé dans la programmation informatique, spécifiquement pour le langage Python. Il est développé par la société tchèque JetBrains. Il fournit l'analyse de code, un débogueur graphique, un testeur d'unité intégré, l'intégration avec des systèmes de contrôle de version (VCS) et prend en charge le développement Web avec Django ainsi que la science des données avec Anaconda

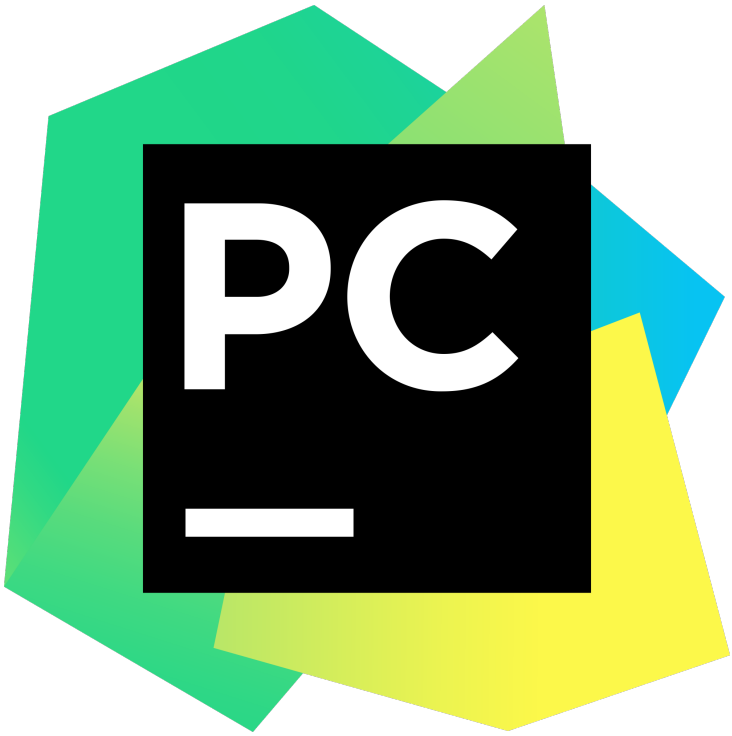

FIG. 3.4 : Logo de PyCharm

3.6 Modèle utilisé pour détection des objets :

3.6.1 YOLOv8

YOLOv8 est un modèle de pointe et à la pointe de la technologie qui s'appuie sur le succès des versions précédentes de YOLO et introduit de nouvelles fonctionnalités et améliorations pour augmenter encore les performances et la flexibilité. YOLOv8 est conçu pour être rapide, précis et facile à utiliser, ce qui en fait un excellent choix pour une large gamme de tâches de détection et de suivi d'objets, de segmentation d'instances, de classification d'images et d'estimation de la pose.

3.7 Méthode de segmentation :

Les étapes de cette méthode sont :

Nous allons considérer une image sous forme de matrice, comme illustré dans la figure3.5.

212	40	255	216	117	151	106	185	66	168	74
109	217	223	251	229	20	206	181	1	171	51
29	47	246	164	200	125	8	37	187	125	158
167	134	7	8	8	236	227	54	201	206	242
255	179	9	10	11	144	157	61	81	238	40
200	73	8	7	9	134	231	186	6	61	29
132	180	6	5	7	64	109	24	31	37	216
215	199	7	10	11	242	70	164	234	151	52
76	31	9	7	8	199	32	177	139	57	10
7	233	119	23	180	135	75	55	125	176	197
44	185	36	239	213	45	222	158	26	107	74
43	86	49	176	96	129	53	194	16	66	234
157	226	2	156	164	244	143	55	177	14	243
49	245	7	75	232	157	251	118	182	148	227
239	14	77	179	162	18	121	93	122	216	72
203	66	144	241	217	247	162	179	123	84	152
24	161	37	147	22	228	42	78	48	2	73
121	78	208	168	16	59	231	107	37	148	18

FIG. 3.5 : Image en tant que matrice

on va decomposer la matrice par des sous matrice de taille 3x3 et on va commancer par la sous matrice cible qui été en bleu, comme nous avons voir dans image ci dessous :

212	40	255	216	117	151	106	185	66	168	74
109	217	223	251	229	20	206	181	1	171	51
29	47	246	164	200	125	8	37	187	125	158
167	134	7	8	8	236	227	54	201	206	242
255	179	9	10	11	144	157	61	81	238	40
200	73	8	7	9	134	231	186	6	61	29
132	180	6	5	7	64	109	24	31	37	216
215	199	7	10	11	242	70	164	234	151	52
76	31	9	7	8	199	32	177	139	57	10
7	233	119	23	180	135	75	55	125	176	197
44	185	36	239	213	45	222	158	26	107	74
43	86	49	176	96	129	53	194	16	66	234
157	226	2	156	164	244	143	55	177	14	243
49	245	7	75	232	157	251	118	182	148	227
239	14	77	179	162	18	121	93	122	216	72
203	66	144	241	217	247	162	179	123	84	152
24	161	37	147	22	228	42	78	48	2	73
121	78	208	168	16	59	231	107	37	148	18

FIG. 3.6 : La sous matrice cible

Ensuite, nous allons comparer le résultat de corrélation de cette sous-matrice cible avec les sous-matrices voisines situées en haut, à gauche et en haut à gauche, comme illustré dans la figure 3.7 où elles sont marquées en vert.

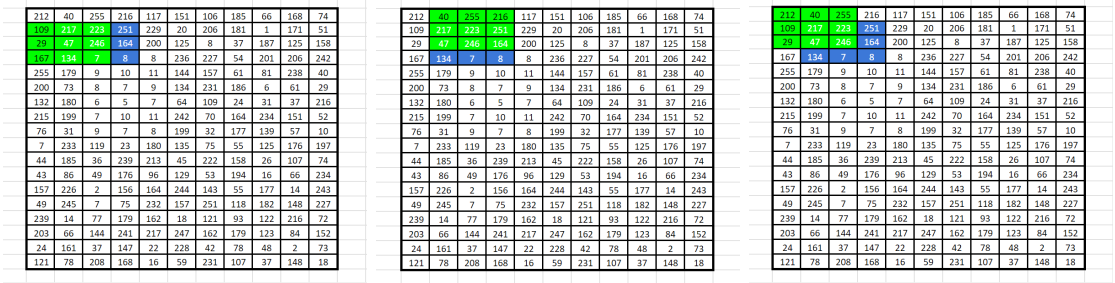

FIG. 3.7 : Les sous-matrices voisines

Si la moyenne du résultat de corrélation de la sous-matrice cible avec ces sous-matrices voisines se rapproche de la valeur de 1 ou -1, cela signifie que ces pixels sont presque identiques. Dans ce cas, nous allons créer une nouvelle matrice vide et Et nous le remplissons (voir la figure 3.8).

Méthode de calcule de nouvelle matrice vide :

Voici la sous-matrice cible :

$$\begin{bmatrix} 221 & 214 & 156 \\ 120 & 17 & 227 \\ 231 & 7 & 8 \end{bmatrix}$$

Voici la sous-matrices voisine qui situées en haut :

$$\begin{bmatrix} 246 & 41 & 230 \\ 221 & 214 & 156 \\ 120 & 17 & 227 \end{bmatrix}$$

Voici la sous-matrices voisine qui situées à gauche :

$$\begin{bmatrix} 24 & 221 & 214 \\ 26 & 120 & 17 \\ 177 & 231 & 7 \end{bmatrix}$$

ET voici la sous-matrices voisine qui situées en haut à gauche :

$$\begin{bmatrix} 91 & 246 & 41 \\ 24 & 221 & 214 \\ 26 & 120 & 17 \end{bmatrix}$$

- On va calculer la résultat de corrélation de sous-matrice cible individuellement avec tous les sous-matrice voisines : (voir section 3.7.1 pour avoir comment calculer la corrélation entre deux matrices)

  La résultat de corrélation avec la sous-matrices voisine qui situées en haut est : **-0.032**

  La résultat de corrélation avec la sous-matrices voisine qui situées à gauche est : **-0.01**

  La résultat de corrélation avec la sous-matrices voisine qui situées en haut à gauche est : **0.11**

- On va calculer la moyenne des résultat de corrélation :

  La moyenne des résultat de corrélation est : **[(-0.032) + (-0.01) + (0.11)]/3 = 0.021**

  Si la valeur absolue de la moyenne des résultat de corrélation est supérieure à **0.6** donc la case qui été en jaune dans la figure 3.8 va remplir comme ceci :

  - Prendre la valeur qui été au centre de sous matrice cible **(246)**.
  - Calculer la somme de cette valeur et les valeurs voisines qui situées en haut, à gauche et en haut à gauche **(246 + 223 + 217 + 43)**.

- Calculer la moyenne des valeurs **(246 + 223 + 217 + 43) / 4 = 143**.
- Finalement, marquer la moyenne dans la case jaune.

Si la valeur absolue de la moyenne des résultat de corrélation est inférieure à **0.6** :

- Marquer directement la valeur qui été au centre de sous matrice cible **(246)**.

143	153	163	166	158	153	165	132	0
93	64	101	160	186	150	186	169	0
72	8	9	64	133	130	119	149	0
41	8	9	28	114	148	123	143	0
81	6	7	67	124	100	111	112	0
62	7	8	82	129	82	95	103	0
57	8	9	39	86	118	153	155	0
99	42	82	66	19	103	142	166	0
79	79	143	118	50	83	89	150	0
101	129	159	127	117	109	94	142	0
122	163	201	158	154	176	176	165	0
145	172	201	171	152	162	165	177	0
210	168	146	150	118	91	69	106	0
179	134	69	132	151	83	43	56	0
79	114	76	101	174	110	92	80	0
0	0	0	0	0	0	0	0	0

FIG. 3.8 : La nouvelle matrice vide après le remplissage

Nous appliquons cette méthode sur toutes les sous-matrices de la matrice vue dans la figure 3.5.

3.7.1 Calculer la corrélation entre deux matrices

Pour calculer la corrélation entre deux matrices mathématiquement, vous pouvez suivre les étapes suivantes :

- Calculez la matrice de covariance : Pour chaque paire de variables dans les deux matrices, calculez la covariance entre ces variables. La covariance mesure la relation linéaire entre deux variables.

Soit X et Y deux matrices de taille n x m. Vous pouvez calculer la matrice de covariance en utilisant la formule suivante :

$$Cov(X,Y) = \frac{1}{n-1}(X - \bar{X})(Y - \bar{Y})^T$$

Où **Cov(X,Y)** est la matrice de covariance, **X** et **Y** sont les matrices d'entrée, **n** est le nombre d'échantillons et **m** est le nombre de variables.

- Calculez les matrices de variance : Calculez les matrices de variance pour chaque matrice d'entrée. La variance mesure la dispersion des valeurs d'une variable.

  Pour une matrice **X** de taille n x m, vous pouvez calculer la matrice de variance en utilisant la formule suivante :

$$Var(X) = \frac{1}{n-1}(X - \bar{X})(X - \bar{X})^T$$

  Où **Var(X)** est la matrice de variance, **X** est la matrice d'entrée, n est le nombre d'échantillons et m est le nombre de variables.

- Calculez la matrice de corrélation : Divisez la matrice de covariance par le produit élément par élément des matrices de variance pour obtenir la matrice de corrélation.

  Pour une matrice de covariance **Cov(X,Y)** et les matrices de variance **Var(X)** et **Var(Y)**, vous pouvez calculer la matrice de corrélation en utilisant la formule suivante :

$$\rho = \frac{\text{Cov}(X,Y)}{\sqrt{\text{Var}(X) \cdot \text{Var}(Y)}}$$

  Où ρ est la matrice de corrélation.

  La matrice de corrélation sera une matrice carrée de taille m x m, où chaque élément ρ(i,j) représente la corrélation entre les variables i et j.

3.8 Implémentation de l'application :

3.8.1 Entraîner YOLOv8 sur des données personnalisées :

Pour entraîner YOLOv8 sur des données personnalisées, nous pouvons suivre les étapes suivantes :

- **Préparation des données :** Collectez et préparez un ensemble de données annotées contenant des images avec des objets que vous souhaitez détecter. Les annotations doivent inclure les coordonnées des boîtes englobantes autour des objets d'intérêt. et ajouter au google drive.

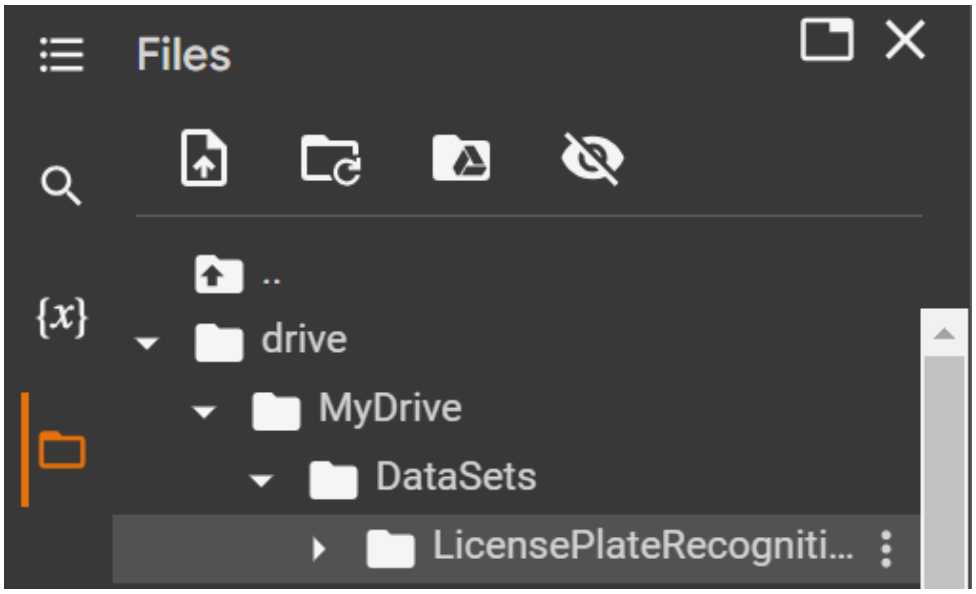

FIG. 3.9 : Dataset location

- **Connecter au google drive :** monter sur /content/drive

```
from google.colab import drive
drive.mount('/content/drive')
```

- **Modifier le fichier data.yaml :** Après d'ouvrir le fichier data.yaml, On va modifier les trois premiers lignes on donne le chemins d'access de dossier train, val et test.

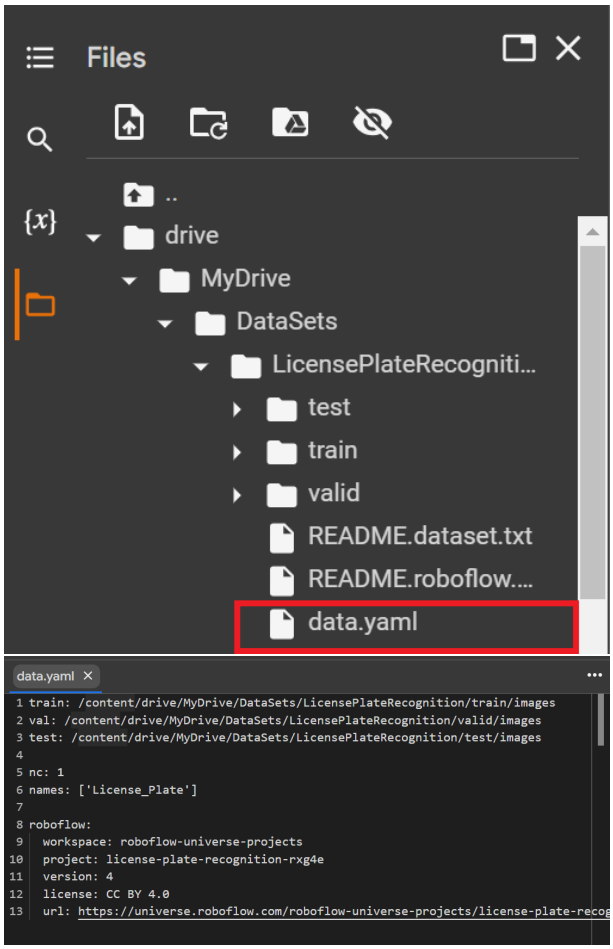

FIG. 3.10 : Modifier le fichier data.yaml

- **Configuration du modèle :**

  Installer Ultralytics library

```
!pip install ultralytics
```

  Importer YOLOv8

```
from ultralytics import YOLO
```

- **Entraînement :**

```
!yolo task=detect mode=train model=yolov8s.pt
data=/content/drive/MyDrive/DataSets/LicensePlateRecognition/data.yaml
epochs=10 imgsz=640 batch=8
project=/content/drive/MyDrive/yolov8/training_results name=LicensePlate
```

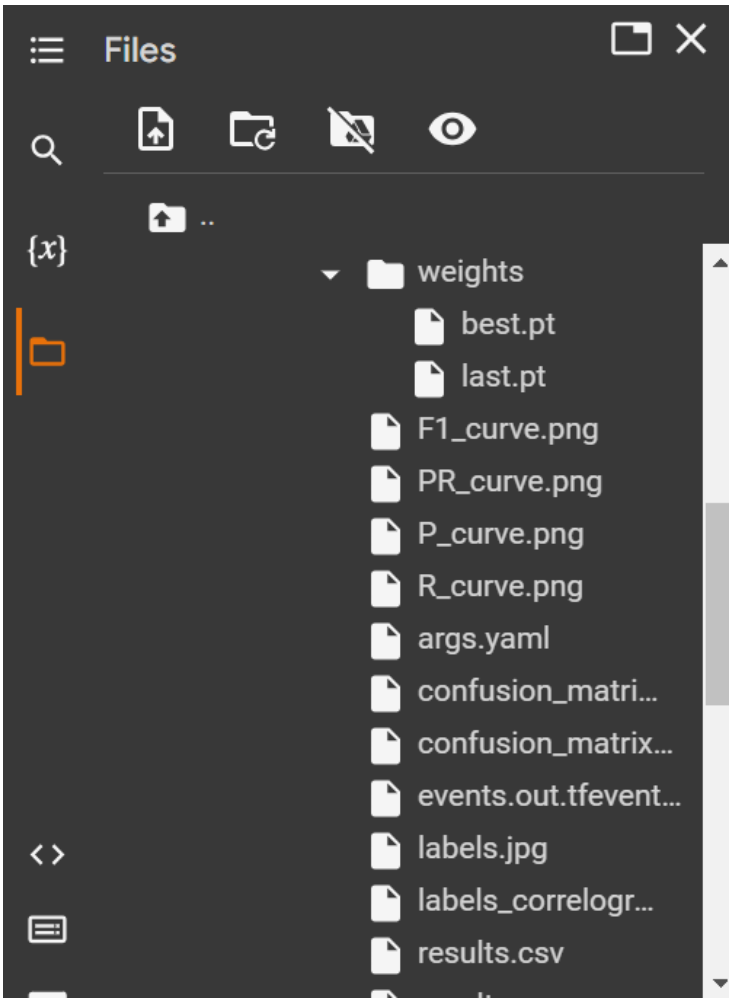

FIG. 3.11 : Résultat de train

Le fichier best.pt c'est la meilleur résultat de train et ce fichier qui on va travailler avec lui.

- **Inférence :**

```
!yolo task=detect mode=predict model=/content/drive/MyDrive/yolov8/
training_results/LicensePlate/weights/best.pt conf=0.55
source=/content/drive/MyDrive/yolov8/test_images
```

- **Exporter le modèle :** Exportez un modèle YOLOv8 vers n'importe quel format pris en charge avec l'argument format, c'est-à-dire format=onnx.

```
!yolo mode=export model=/content/drive/MyDrive/yolov8/training_results/
LicensePlate/weights/best.pt format=onnx
```

3.8.2 Image d'origine :

FIG. 3.12 : Image d'origine

3.8.3 La fonction process_license_plate()

```
def process_license_plate(img,coordinates) :

    # Extract license plate region from the image based on given coordinates
    x,y,w,h = int(coordinates[0]), int(coordinates[1]), int(coordinates[2]),
              int(coordinates[3])
    img = img[y :h,x :w]

    # Convert region to grayscale
    license_plate_gray = cv2.cvtColor(img , cv2.COLOR_RGB2GRAY)
    license_plate_gray = cv2.resize(license_plate_gray, None, fx = 3, fy = 3,
                                    interpolation = cv2.INTER_CUBIC)
```

Explication :

Cette fonction traite une image de plaque d'immatriculation. Les coordonnées de la région de la plaque d'immatriculation sont fournies en tant que paramètre '**coordinates**'.

- La ligne 1 déclare la fonction '**process_license_plate**' qui prend deux paramètres : '**img**' (l'image d'entrée) et '**coordinates**' (les coordonnées de la région de la plaque d'immatriculation dans l'image).
- La ligne 4 extrait les valeurs de **x**, **y**, **w** et **h** à partir des coordonnées fournies. Ces valeurs sont converties en entiers à l'aide de la fonction '**int()**'.
- La ligne 6 découpe la région de la plaque d'immatriculation de l'image en utilisant les coordonnées fournies. La variable '**img**' est modifiée pour ne contenir que la région d'intérêt.
- Les lignes 9 et 10 convertissent la région de la plaque d'immatriculation en niveaux de gris et la redimensionnent. La fonction '**cv2.cvtColor()**' convertit l'image en niveaux de gris en utilisant le code de couleur '**cv2.COLOR_BGR2GRAY**. Ensuite, la fonction '**cv2.resize()**' redimensionne l'image en utilisant des facteurs d'échelle de 3 dans les deux dimensions ('**fx=3** et '**fy=3**).
- L'interpolation '**cv2.INTER_CUBIC** est utilisée pour effectuer le redimensionnement.

Ce code permet de préparer la région de la plaque d'immatriculation pour d'autres traitements, tels que la reconnaissance des caractères ou d'autres opérations spécifiques à l'analyse des plaques d'immatriculation.

Résultat obtenu :

FIG. 3.13 : Région de la plaque d'immatriculation en niveaux de gris

3.8.4 La fonction de segmentation()

```
def segmentation(license_plate) :

    # Convert the grayscale image to a matrix
    matrix = np.array(license_plate)

    num_rows = matrix.shape[0]
    num_columns = matrix.shape[1]

    # Initialize the new matrix
    new_matrix = np.zeros((num_rows - 2, num_columns - 2), dtype=int)

    # Perform the operation on the matrix
    for row in range(2, num_rows - 1) :
        for column in range(2, num_columns - 1) :
            # Get the submatrix
            submatrix = matrix[row - 1 :row + 2, column - 1 :column + 2]

            # Check if the submatrix contains valid values
            if np.isnan(submatrix).any() or not np.isfinite(submatrix).all() :
                new_matrix[row - 2, column - 2] = matrix[row, column]
            else :
                # Compute the correlations
                top_correlation = np.corrcoef(submatrix.ravel(),
                    matrix[row - 2 :row + 1, column - 1 :column + 2].ravel())
                left_correlation = np.corrcoef(submatrix.ravel(),
                    matrix[row - 1 :row + 2, column - 2 :column + 1].ravel())
                top_left_correlation = np.corrcoef(submatrix.ravel(),
                    matrix[row - 2 :row + 1, column - 2 :column + 1].ravel())

                # Compute the average correlation
                average_correlation = (top_correlation[0, 1] +
                    left_correlation[0, 1] +
                    top_left_correlation[0, 1]) / 3

                # Compute the average of the values
                voisins = int(np.mean(matrix[row - 1][column] +
                    matrix[row - 1][column - 1] +
                    matrix[row][column - 1]))
                my_position = int(np.mean(matrix[row][column]))
                average_value = (my_position + voisins) / 4

                if np.abs(average_correlation) < 0.6 :
                    new_matrix[row - 2, column - 2] = average_value
```

```
                else :
                    new_matrix[row - 2, column - 2] = matrix[row, column]

    # Convert the new matrix back to an image
    new_license = new_matrix.astype(np.uint8)

    return new_license
```

Explication :

La fonction '**segmentation** effectue la segmentation d'une image de plaque d'immatriculation.

- La fonction prend en entrée '**license_plate**, qui est une image de la plaque d'immatriculation quand appliquer le seuillage après de convertie en niveaux de gris.
- La matrice '**matrix** est créée à partir de l'image en niveaux de gris.
- Les dimensions de la matrice sont obtenues pour déterminer le nombre de lignes et de colonnes.
- Une nouvelle matrice, '**new_matrix**, est initialisée avec des dimensions réduites pour stocker les résultats de la segmentation.
- Une double boucle '**for**' est utilisée pour parcourir chaque pixel de la matrice d'origine à partir des indices 2 jusqu'à '**num_rows - 1**' pour les lignes et de 2 jusqu'à '**num_columns - 1**' pour les colonnes.
- Pour chaque pixel, une sous-matrice 3x3 est extraite de la matrice d'origine.
- On vérifie si la sous-matrice contient des valeurs invalides, telles que des valeurs NaN (non-numériques) ou non finies. Si c'est le cas, le pixel correspondant dans la nouvelle matrice prend la valeur du pixel d'origine.
- Sinon, on calcule les corrélations entre la sous-matrice et différentes régions voisines de l'image d'origine.
- La corrélation moyenne est calculée en prenant la moyenne des corrélations obtenues à partir des différentes régions voisines.
- On calcule également la moyenne des valeurs des pixels voisins et du pixel actuel pour obtenir une valeur moyenne.
- Si la valeur absolue de la corrélation moyenne est inférieure à **0.6**, le pixel correspondant dans la nouvelle matrice prend la valeur moyenne calculée. Sinon, il conserve la valeur du pixel d'origine.
- Une fois que tous les pixels ont été traités, la nouvelle matrice est convertie en une image en utilisant le type de données '**uint8**' pour les valeurs des pixels.

- L'image segmentée est renvoyée en tant que résultat de la fonction.

Ce code réalise une segmentation de l'image de la plaque d'immatriculation en utilisant des opérations basées sur les valeurs des pixels et les corrélations avec les pixels voisins.

Résultat obtenu :

FIG. 3.14 : Après segmentation

3.8.5 Appliquer un seuillage :

```
#Apply thresholding
    _, license_plate_binary = cv2.threshold(license_plate_gray, 70, 255,
                                            cv2.THRESH_BINARY | cv2.THRESH_OTSU)
```

Cette ligne de code effectue un seuillage sur l'image '**license_plate_gray** en utilisant la fonction '**cv2.threshold()**' de la bibliothèque OpenCV.

- Le premier paramètre, '**license_plate_gray**', est l'image sur laquelle le seuillage est appliqué.
- Le deuxième paramètre, '**70**, est la valeur de seuil. Tous les pixels ayant une valeur inférieure à ce seuil seront convertis en noir (0), tandis que les pixels ayant une valeur supérieure ou égale au seuil seront convertis en blanc (255).

- Le troisième paramètre, '**255**', est la valeur maximale à utiliser pour les pixels supérieurs au seuil.
- Le quatrième paramètre, '**cv2.THRESH_BINARY**, est un drapeau indiquant le type de seuillage à appliquer. Dans ce cas, '**cv2.THRESH_BINARY**.
- Le quatrième paramètre, '**cv2.THRESH_OTSU**' est un type de seuillage plus avancé qui détermine automatiquement une valeur de seuil optimale en fonction de l'histogramme de l'image. Il peut être utile lorsque la valeur de seuil n'est pas connue à l'avance ou lorsqu'il y a des variations d'éclairage.

Le résultat de cette opération est stocké dans la variable '**license_plate_binary**, qui contient l'image seuillée de la plaque d'immatriculation.

Résultat obtenu :

FIG. 3.15 : Après d'appliquer un seuillage

3.8.6 La lecture de la plaque :

```
    # Initialize EasyOCR reader for English language with GPU support
    reader = easyocr.Reader(['en'], gpu=True)

    new_license = segmentation(license_plate_binary)

    result = reader.readtext(new_license)

    license_plate_texts = []

    for res in result :
        if len(result) == 1 :
            license_plate_texts.append(res[1])
        if len(result) > 1 and len(res[1]) > 6 and res[2] > 0.6 :
            license_plate_texts.append(res[1])

```

```
    # Apply additional filtering or post-processing to eliminate duplicates
    unique_license_plate_texts = list(set(license_plate_texts))

    return " ".join(unique_license_plate_texts).upper()
```

Explication :

Ce code utilise la bibliothèque EasyOCR pour reconnaître le texte sur l'image seuillée de la plaque d'immatriculation.

- La variable reader est initialisée en utilisant '**easyocr.Reader(['en'], gpu=True)**. Cela crée un lecteur EasyOCR qui utilise le modèle de reconnaissance optique de caractères (OCR) pour la langue anglaise avec prise en charge GPU activée.
- La méthode '**reader.readtext(license_plate_binary)** est utilisée pour lire le texte dans l'image seuillée de la plaque d'immatriculation. Le résultat est stocké dans la variable result, qui contient une liste de tuples contenant le texte, les coordonnées et la confiance de chaque détection de texte.
- Une boucle for est utilisée pour parcourir chaque élément du résultat de la lecture du texte.
- Si la liste de résultats ne contient qu'un seul élément, le texte est ajouté à la liste '**license_plate_texts**'.
- Si la liste de résultats contient plusieurs éléments et que le texte a une longueur supérieure à 6 caractères et une confiance supérieure à **0.2**, le texte est ajouté à la liste '**license_plate_texts**'.
- Ensuite, une étape de filtrage ou de post-traitement supplémentaire est appliquée pour éliminer les doublons dans la liste '**license_plate_texts**'. Les textes uniques sont stockés dans la liste '**unique_license_plate_texts**'.
- Enfin, les textes de plaque d'immatriculation uniques sont concaténés en une seule chaîne, convertis en majuscules et renvoyés en tant que résultat de la fonction.

Résultat obtenu :

FIG. 3.16 : Lecture des caractères

3.8.7 Résultat obtenu sur quelque autres vehicules :

A)

Fig. 3.17 : Image originale

FIG. 3.18 : A gauche avec la segmentation et A droite sans segmentation

Desscussion sur la resultat :

Comme nous avons voir dans la figure 3.18 :

- **Avec la segmentation :** La lecture de plaque est 100% juste
- **Sans segmentation :** Il y a une seule faute dans un seul caractère, donc la lecture de plaque est 85.71% juste

B)

FIG. 3.19 : Image originale

FIG. 3.20 : A gauche avec la segmentation et A droite sans segmentation

Desscussion sur la resultat :

Comme nous avons voir dans la figure 3.20 :

- **Avec la segmentation :** La lecture de plaque est 100% juste
- **Sans segmentation :** La lecture de plaque est 100% juste

C)

FIG. 3.21 : Image originale

FIG. 3.22 : A gauche avec la segmentation et A droite sans segmentation

Desscussion sur la resultat :

Comme nous avons voir dans la figure 3.22 :

- **Avec la segmentation :** La lecture de plaque est presque 100% juste, sauf qu'il ajoute une point virgule (;) à la fin.
- **Sans segmentation :** Il y a une seule faute dans un seul caractère (remplace la première W par V), donc la lecture de plaque est 85.71% juste

D)

FIG. 3.23 : Image originale

FIG. 3.24 : A gauche avec la segmentation et A droite sans segmentation

Desscussion sur la resultat :

Comme nous avons voir dans la figure 3.24 :

- **Avec la segmentation :** La lecture de plaque est 100% juste
- **Sans segmentation :** La lecture de plaque est 100% juste

E)

FIG. 3.25 : Image originale

FIG. 3.26 : A gauche avec la segmentation et A droite sans segmentation

Desscussion sur la resultat :

Comme nous avons voir dans la figure 3.26 :

- **Avec la segmentation :** La lecture de plaque est 100% juste
- **Sans segmentation :** Il y a une seule faute dans un seul caractère (Il ajoute 4 entre 16 et 122), donc la lecture de plaque est 85.71% juste

F)

FIG. 3.27 : Image originale

FIG. 3.28 : A gauche avec la segmentation et A droite sans segmentation

Desscussion sur la resultat :

Comme nous avons voir dans la figure 3.28 :

- **Avec la segmentation :** La lecture de plaque est 100% juste
- **Sans segmentation :** La lecture de plaque est 100% juste

G)

FIG. 3.29 : Image originale

FIG. 3.30 : A gauche avec la segmentation et A droite sans segmentation

Desscussion sur la resultat :

Comme nous avons voir dans la figure 3.30 :

- **Avec la segmentation :** La lecture de plaque est 100% juste
- **Sans segmentation :** La lecture de plaque est 100% juste

H)

FIG. 3.31 : Image originale

FIG. 3.32 : A gauche avec la segmentation et A droite sans segmentation

Desscussion le résultat :

Comme nous avons voir dans la figure 3.32 :

- **Avec la segmentation :** La lecture de plaque est presque 100% juste, sauf qu'il ajoute 5 à la fin.
- **Sans segmentation :** Il y a une seule faute dans un seul caractère, donc la lecture de plaque est presque 100% juste , sauf qu'il ajoute 5 à la fin.

Nous remarquons qu'il y a pas beaucoup des erreurs après la segmentation lorsque la qualité de l'image est bonne.

3.9 Conclusion

Dans ce chapitre, nous avons décrit la méthodologie utilisée pour mettre en place un système de reconnaissance automatique des plaques d'immatriculation. Nous avons détaillé les résultats obtenus pour les trois modules essentiels : localisation, segmentation et reconnaissance. Ensuite, nous avons procédé à une analyse et une discussion des résultats, en mettant en évidence les problèmes rencontrés et en proposant des solutions pour améliorer notre travail.

Conclusion générale

La circulation routière devient de plus en plus dense dans notre vie quotidienne. Afin de gérer plus efficacement le trafic, il est primordial d'adopter de nouvelles technologies, telles que des caméras de surveillance et des systèmes de sécurité. La plaque d'immatriculation joue un rôle essentiel en tant que moyen d'identification des véhicules. L'exploitation des informations extraites des images ou des vidéos provenant de ces plaques facilite considérablement diverses opérations et processus, notamment dans le domaine de la sécurité, ainsi que la classification automatique des données qu'elles contiennent.

Nous avons développé dans ce projet une technique de lecture automatique des caractères et des plaques d'immatriculation à partir d'images numériques. Cette technique est basée sur une nouvelle approche de segmentation des caractères. Pour cela, nous avons utilisé diverses fonctionnalités de la bibliothèque OpenCV, ce qui nous a permis de détecter, localiser, filtrer, segmenter et identifier les caractères grâce à la reconnaissance optique des caractères.

L'implémentation de ce système nous a offert l'opportunité d'approfondir nos connaissances théoriques et pratiques. Nous avons pu nous familiariser avec certains concepts fondamentaux dans le vaste domaine du traitement d'images, tout en appliquant les connaissances acquises lors de nos études universitaires.

Il est important de souligner que ce système, développé tout au long de ce projet, nécessite une maintenance régulière et des ajustements afin de s'adapter aux nombreux changements qui peuvent survenir.

Pour poursuivre, nous pourrions envisager d'améliorer notre système en optimisant les algorithmes de segmentation et en augmentant la précision de la reconnaissance optique des caractères. De plus, nous pouvons adapter notre système aux besoins spécifiques en l'intégrant dans des applications de surveillance de la circulation, de gestion du stationnement ou de contrôle d'accès. Explorer d'autres domaines du traitement d'images tels que la détection d'objets ou la reconnaissance faciale pourrait également être une option intéressante. Enfin, nous pouvons partager nos résultats, rédiger des articles techniques, présenter notre travail lors de conférences et collaborer avec d'autres experts pour élargir nos connaissances et participer à des projets de recherche avancés.

Bibliographie

[2] Lucas J. van Vliet Ian T. Young Jan J. Gerbrands. "Fundamentals of Image Processing". In : *Delft University of Technology* 112 (2007), p. 3.

[7] Mohamed Sandeli. "Analyse d'image basée sur les réseaux de neurones profonds". Thèse de doct. Université Constantine 2, 2021.

[8] Souhila Guerfi Ababsa. "Authentification d'individus par reconnaissance de caractéristiques biométriques liées aux visages 2D/3D". Thèse de doct. Université d'Evry Val d'Essonne, 2008.

[9] Patrick J. Flynn Kyong I. Chang Kevin W. Bowyer. "An evaluation of multimodal 2D+3D face biometrics". In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.4 (2005), p. 619-624.

[10] Maria Petrou et Pedro García-Sevilla. *Image Processing : Dealing With Texture*. Mai 2006. isbn : 978-0-470-02628-1.

[11] C H Chen, L F Pau et P S P Wang. *Handbook of Pattern Recognition and Computer Vision*. WORLD SCIENTIFIC, 1993.

[12] P. Brodatz. *Textures : A Photographic Album for Artists and Designers*. Dover books on art, graphic art, handicrafts. Dover Publications, 1966. isbn : 9780486216690.

[14] Saining Xie et al. "Aggregated Residual Transformations for Deep Neural Networks". In : *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), p. 5987-5995.

[15] Tomasz Malisiewicz, Abhinav Gupta et Alexei A. Efros. "Ensemble of Exemplar-SVMs for Object Detection and Beyond". In : *ICCV*. 2011.

[16] Ross B. Girshick. "Fast R-CNN". In : *2015 IEEE International Conference on Computer Vision (ICCV)* (2015), p. 1440-1448.

[18] Zhipeng Deng et al. "Multi-scale object detection in remote sensing imagery with convolutional neural networks". In : *ISPRS Journal of Photogrammetry and Remote Sensing* 145 (mai 2018).

[20] Joseph Redmon et al. "You Only Look Once : Unified, Real-Time Object Detection". In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2016, p. 779-788.

Webographie

[1] A. Bertrand. *Lapi ou VPI quelles différences*. Dernière visite le 12/06/2023. 2018. URL : http://cameravideosurveillancesecurite.blogspot.com/p/lapi-ou-vpi-quelles-differences.html.

[3] Printdeal. *RGB / CMYK : définition et différences*. https://www.printdeal.be/fr/blog/show/tutoriaux/rgb-cmyk-definition-et-differences. Consulté le 23 juin 2023. date de consultation.

[4] *Vidéo et imagerie numérique - Traitement numérique d'images*. fr. Dernière visite le 11/02/2023. Université de New South Wales, année. URL : https://web.maths.unsw.edu.au/~lafaye/CCM/video/traitimg.htm.

[5] Wikipédia. *Lissage d'images — Wikipédia, l'encyclopédie libre*. [En ligne ; Page disponible le 15-janvier-2021]. 2021. URL : http://fr.wikipedia.org/w/index.php?title=Lissage_d%27images&oldid=178819645.

[6] Wikipédia. *Détection de contours — Wikipédia, l'encyclopédie libre*. [En ligne ; Page disponible le 25-janvier-2019]. 2019. URL : http://fr.wikipedia.org/w/index.php?title=D%C3%A9tection_de_contours&oldid=156146784.

[13] MathWorks. *Object Recognition*. https://fr.mathworks.com/solutions/image-video-processing/object-recognition.html. Dernière visite le 12/06/2023. 2023.

[17] Paperspace. *Faster R-CNN Explained : Object Detection*. https://blog.paperspace.com/faster-r-cnn-explained-object-detection/. Consulté le 23 juin 2023. date de consultation.

[19] *Ultralytics Documentation*. Dernière visite le 12/06/2023. URL : https://docs.ultralytics.com/.

[21] Wikipédia. *OpenCV — Wikipédia, l'encyclopédie libre*. [En ligne ; Page disponible le 2-octobre-2022]. 2022. URL : http://fr.wikipedia.org/w/index.php?title=OpenCV&oldid=197436603.