REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE YAHIA FARES DE MEDEA



Faculté de **T**echnologie

Mémoire de fin d'études de Master

Filière : Génie électrique
Spécialité : Electronique des Système embarqué

Gestion d'un Parking par vision artificiel

Présenté par :

BELHADJ MOHAMD MOKHTAR.

RENANE MOHAMED.

Proposé et Dirigé par :

Mr.ESSAID BILLAL

Promotion 2019

Remerciements

En préambule à ce mémoire, nous remercions ALLAH qui nous a aidé et donné la patience et le courage durant ces longues années d'étude.

Nous souhaitons adresser nos remerciements les plus sincères aux personnes qui nous ont apporté leur aide et qui ont contribué à l'élaboration de ce mémoire.

Ces remerciements vont aussi au corps professoral et administratif de la Faculté des Sciences de l'Université de medea, pour la richesse et la qualité de leur enseignement et qui déploient de grands efforts pour assurer à leurs étudiants une formation actualisée.

Nous adressons un grand remerciement à notre encadreur Mr.Essaid bilal pour nous avoir proposé ce projet de fin d'étude, pour ses conseils et ses directives du début à la fin de ce travail.

Nos vifs remerciements vont également aux membres du jury pour l'intérêt qu'ils ont porté à notre projet en acceptant d'examiner ce travail.

Nous nous ne remercierons jamais assez, nos chers parents pour leur amour, leur soutien et leur Patience.

Enfin, nous adressons nos plus sincères remerciements à tous nos proches et amis, qui nous ont toujours encouragés au cours de la réalisation de ce mémoire.

Merci à tous et à toutes.

Dédicaces

Je dédie ce mémoire
A mes chers parents ma mère et mon père pour leur patience, leur amour, leur soutien et leurs encouragements.
A mes frère ishak et abderhman mes soeurs lamia et bouchra.
En témoignage de la fraternité, Avec mes souhaits de bonheur de santé et de succès.
A tous mes amis, mes camarades et mes collègues de promotion avec qui j'ai passé mes meilleures années d'études.

BELHADJ MOHAMED MOKHTAR.

Dédicace

Au nom d'Allah le miséricordieux

Je dédie ce modeste travail

A mes chers parents, mes chères frères famille

A tous mes amis.

A tous ceux qui m'ont aide durant ma vie Universitaire.

RENANE MOHAMED.

ملخص :

الهدف من هذه المذكرة هواستخدام نظام جديد للتعرف على لوحات التسجيل على أساس (كا ان ان).

تم تشكيل نظامنا من ثلاث مراحل رئيسية، وهي مرحلة المعالجة المسبقة ، والتي تشمل مرحلة الكشف عن اللوحة، مرحلة التجزئة لعزل الحروف التي أنشأها. المحور الثالث في نظامنا يمثل مرحلة التعرف القائم على أساليب التصنيف المراقب. وقد طبقت هذه المرحلة على قاعدة بيانات لوحات الترقيم التي تظهر نتائج التجارب المتحصل عليها و التي حققت نسبة تعرف جد عالية.

الكلمات الأساسية: التعرف على لوحة الأرقام، نظام ال بي ار، استخراج الميزات، التجزئة،

الأقرب للجهاز المجاور. K-NN

Résumé :

L'objectif de ce mémoire est d'utiliser un nouveau système de reconnaissance des plaques d'immatriculation basé sur k-NN le plus proche voisin. Notre système est constitué de trois étapes majeures qui sont la phase de prétraitements, qui inclue une étape de la détection de la plaque et une étape de segmentation pour isoler les caractères Le troisième axe constitue la phase de la reconnaissance basée dans notre système sur des méthodes de classification supervisée. Cette étape été appliquées sur une base de données des plaques d'immatriculation. Les résultats des expérimentations obtenus montrent que la méthode proposée achève des taux de reconnaissance élevés.

Mots clés : La reconnaissance des plaques d'immatriculation, système LPR, extraction des caractéristiques, segmentation, k-NN plus proche voisin.

Abstract:

The goal of this work is to use a new registration plate recognition system based on the nearest neighbor k-NN. Our system consists of three major steps, which are the pre-treatment phase, which includes a plate detection step and a segmentation step to isolate the characters The third axis is the recognition phase based in our system on supervised classification methods. This step was applied on a license plate database. The results of the experiments obtained show that the proposed method is completing high recognition rates.

Keywords: License plate recognition, LPR system, feature extraction, segmentation, k-NN nearest neighbor.

Table des matières

I N T R O D U C T I O N G E N E R A L E **1**

Chapitre I : Généralités des systèmes de reconnaissance.

I.1 Introduction : 5

I.2 Etat de l'art : 5

I.3 Domaine d'utilisation des systèmes de reconnaissance des plaques : 6

I.4 Système LPR : 7

I.5 Organisation générale d'un système de reconnaissance : 8

I.5.1 Phase d'acquisition : 9

I.5.2 Phase de prétraitements : 9

I.5.3 Phase de segmentation : 10

I.5.4 Phase de classification : 10

I.5.5 Etape d'apprentissage : 10

I.6 Systèmes de reconnaissance de caractères OCR : 10

I.6.1 Historique d'O.C.R : 11

I.6.2 Introduction : 11

I.6.3 Définition : 12

I.6.4 Domaine d'application O.C.R : 12

I.6.5 Différents aspects de l'OCR : 13

I.6.6 Classification de l'OCR : 14

I.6.6.1 Reconnaissance en-ligne et hors-ligne : 14

I.6.7 Problèmes liés à l'OCR : 15

Conclusion : 16

C H A P I T R E II : LES APPROCHES DE CLASSIFICATION.

II.1 Introduction : 17

II.2 Approche de classification : 17

II.2.1 Introduction : 17

II.2.2 Définition de la classification : 18

II.2.2.1 Classification non supervisé : 18

II.2.2.2 Classification Semi supervisé : 18

II.2.2.3 Classification Supervisée : 19

II.3 Types des méthodes de classification supervisée : ……19
II.3.1 Réseaux de neurones : Perceptron multicouche : ……..19
II.3.2 K-plus proches voisins : ……21
II.3.2.1 Comment fonctionne l'algorithme KNN? ……...21
II.3.2.2 Exemple de k-NN : ……23
II.3.2.3 L'extension de 1-NN à k-NN se fait comme suit : ……24
II.3.2.4 Les avantages de la méthode des K plus proches voisins on cite [53] : …….25
II.3.2.5 Les inconvénients de la méthode des K plus proches voisins on cite [55] : ..25
II.3.2.6 Reconnaissance de caractères en utilisant KNN : ……26
II.3.3 Support vecteur machine (SVM): ……27
II.3.3.1 Introduction : ……..27
II.3.3.2 1. Pourquoi les Machines à Vecteurs de Support (SVM) ? ……..27
II.3.3.3 Définition : ……..28
II.3.3.4 Les avantages de la méthode SVM : ……29
II.3.3.5 Les inconvénients de la méthode SVM : ……29
II.3.4 Modèles de Markov caches (HMM) : ……30
II.3.4.1 Introduction : ……..30
II.3.4.2 Les trois problèmes des HMMs : ……30
II.3.4.3 Les avantage des HMMs [66] : ……31
II.3.4.4 Les inconvénients des HMMs : ……31
II.3.4.5 Technique utilisé pour La reconnaissance des caractères à l'aide de HMM :.31
II.3.4.6 Étape à laquelle l'algorithme de contour HMM est appliqué en tant que classificateur pour la reconnaissance : ……32
Conclusion : ……..33

Chapitre III : IMPLEMETATION DES SYSTEMES DE LPR.

III.1 Introduction : ……35
III.2 Outils de développement : ……35
III.2.1 Anaconda : Qu'est-ce que Anaconda? ……35
III.2.2 Démarrage d'Anaconda : ……..35
III.2.3 Disposer des dernières mises à jour : ……...36
III.3 Environnement de travail : ……37
III.3.1 Spyder : ……37
III.3.1.1 Pourquoi choisir python comme outil de développement : ……..39

III.3.2 OpenCV : ……39
III.3.3 L'interface graphique (GUI) : ……..41
III.3.4 tkinter library : ……41
III.3.5 Remarque: ……41
III.3.6 Conclusion ……...41
III.4 IMPLEMENTATION ……...42
III.4.1 Introduction :……42
III.4.2 Architecture générale :……42
III.4.3 Fonctionnement : ……43
III.4.3.1 Les interfaces de l'application : ……44
III.4.4 Démarche suivie : ……...45
III.4.4.1 Caméra :……45
III.4.4.2 Couleur ou Noir et Blanc :……45
III.4.4.3 Position de la caméra : ……...46
III.4.5 Étapes de la reconnaissance des plaques d'immatriculation avec python : …...47
III.4.5.1 License Plate Détection : ……..47
III.4.5.2 Character Segmentation: ……48
III.4.5.3 Reconnaissance des caractères: ……48
III.4.6 Architecture et conception d'Ocr : ……...48
III.4.7 La détection des plaques : ……49
a) Prétraitement : ……49
c) La détection des caractères : ……54
v Prétraitement : ……55
4) Étape 4 : Réperté tous les contours de l'image pour supprimer les limites de l'image……..56
III.5 Résultats expérimentaux : ……58
Conclusion :…… 59

Table des Figures :

CHAPITRE I :

Figure I. 1: 6
2Figure I. 2:*Exemple de la position de la camera et la plaque.* 7
Figure I. 3: *Schéma général d'un système de reconnaissance de caractères.* 9
Figure I. 4: *Différents aspects de l'OCR.* 14
Figure I. 5:*Exemple de reconnaissance en ligne* 15

CHAPITRE II :

Figure II. 1 : *Schéma illustrant un perceptron multicouche avec deux couches cachées*…….18
Figure II. 2:*exemple de classification lorsque K=1* ……..20
Figure II. 3:*Les étapes de base de KNN* ……*21*
Figure II. 4: *3eme étape :trouver les voisins.*.....21
Figure II. 5: *Exemple de classification par KNN*……..22
Figure II. 6 : *Illustration du principe des SVMs*……27
Figure II. 7: *structure of hidden states*……28
Figure II. 8: *Montrant ce que le logiciel recherche exactement*……30
Figure II. 9: *traitement de classification de modèle utilisé*……31

CHAPITRE III :

Figure III. 1: *Interface graphique de Anaconda Navigator*……*36*

Figure III. 2 : *Fenetre de Anaconda prompt*……36

Figure III. 3: *Gestionnaire de package*……*36*

Figure III. 4: *Fenêtre console interactive ipython*……*37*

Figure III. 5: *Architecture générale du système*……*41*

Figure III. 6: U*n capture montrant la phase d'entrer au parking*……*42*

Figure III. 7:*Une capture de l'interface montrant la phase de sortie du parking*……*43*

Figure III. 8: *Reconnaissance des plaque de matriculassions*……*44*

Figure III. 9: *Étapes typiques d'un système RAPI*……*45*

Figure III. 10: *L'architecture de notre système OCR*……*47*

Figure III. 11: *Image original RGB*……*48*

Figure III. 12: *Image a niveau de gris*……*49*

Figure III. 13: *Image a niveau de gris*……*49*

Figure III. 14: *Image binaire par la méthode Seuil adaptatif avec moyenne pondérée gaussienne……50*

Figure III. 15: *l'image binaire introduite sur contour founding……51*

Figure III. 16: *Image filtré par rapport au caractère existant……52*

Figure III. 17 : *les caractères détectés de la plaque……53*

Figure III. 18: *image originale coupé……54*

Figure III. 19: *image a niveau de gris……54*

Figure III. 20:image binairisée……..54

Figure III. 21: *Lissage de l'imagepar le filtre Gaussien……54*

Figure III. 22:*Liste de tous les contours dans l'image……55*

Figure III. 23: *Toute la plaque possible qui peut être trouvé……55*

Figure III. 24: *Extracted plate……55*

Figure III.26: *Identification de la plaque d'immatriculation……56*

Figure III. 25: E*xemple d'un échec de reconnaissance……56*

Figure III. 26: *Reconnaissance de plaque réussite……57*

Figure III. 27: *Matricule alphanumérique……58*

CONCLUSION GENERALE *……60*

REFERENCES

WEBGRAPHIE

ANNEXE

Liste des abréviations

OCR : Optical Character Recognition.

CR : Character recognition.

Svm : support Vector machine**.**

Hmm : Hidden markov model

Knn : k- nearest neighbor **.**

LPR : Licence plate recognition.

LP : licence plate.

GUI : Graphic user interface.

ASCII : American standard code for information interchange.

PMC : perceptron multi-layer.

INTRODUCTION GENERALE

Introduction générale:

Contexte :

Le numéro d'immatriculation représente un moyen efficace pour identifier les véhicules. Il s'agit d'une information unique pour chaque voiture.

Fréquemment, il est nécessaire d'identifier les plaques d'immatriculation des véhicules pour la sécurité. Les informations extraites peuvent être utilisées pour plusieurs intérêts, comme le contrôle d'accès et de flux, la surveillance des passages aux frontières et aux péages, la recherche de véhicules suspects ou encore la lutte contre la criminalité, etc.

Ceci rend leurs lectures cruciale et inévitable dans tous ces domaines.

Dans notre projet, nous nous intéressons à la reconnaissance et la lecture automatique des plaques d'immatriculation à partir de captures d'images prises sur le devant ou à l'arrière des véhicules.

Nous avons décomposé notre mémoire comme suit :

Le premier chapitre nous allons présenter ce qu'est un OCR ainsi que le processus utilisé et ses domaines d'application de la reconnaissance des caractères.

Le deuxième chapitre, nous commencerons par définir les approches de classification et les différentes méthodes utilisées pour la reconnaissance des caractères.

Le troisième et dernier chapitre, nous présenterons l'implémentation détaillée de l'application en expliquant les interfaces graphiques de notre système.

Après ceci, nous allons présenter les algorithmes qui doivent être réalisés pour que le logiciel puisse identifier une plaque d'immatriculation et les difficultés qui en résultent.

Nous terminerons ce mémoire par une conclusion générale.

Problématique et objectif :

La vraie problématique pour identifier des plaques d'immatriculation réside dans le fait de pouvoir faire de la reconnaissance optique de caractères sur une petite partie d'image extraite de séquences enregistrées, 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 un prétraitement de ces images à la détection des contours, pour permettre une reconnaissance optique assez fiable des caractères.

Ce que nous visons à travers notre travail est de faciliter la tâche d'identification des caractères du matricule en exploitant les avantages que peux offrir le traitement d'image.

Notre approche :

On propose un système de reconnaissance des plaque d'immatriculation basé sur la méthode de classification k-NN On propose de détection des caractères et les chiffre en passant par une première phase, qui est la détection de la plaque d'immatriculation Pour cela, nous avons développé une application qui consiste à réaliser des prétraitements, Et la détection et la lecture des chiffres et des caractères. Nous besoin d'utiliser certaines des fonctions pour faciliter le problème pour reconnaitre l'objet (caractère, chiffre).

CHAPITRE I

GENERALITES DES SYSTEMES DE RECONNAISANCE.

Chapitre I :

I.1 Introduction :

Nous commencerons ce chapitre par une définition des systèmes de reconnaissance, sa généralité et son objectif. Après ceci, nous allons présenter la technique de la reconnaissance optique de caractères, ces outils, domaine d'application O.C.R et les problèmes liées à l'O.C.R. Nous finirons ce chapitre par citer ces principaux modules.

Le but de la reconnaissance des plaques d'immatriculations est d'identifier les véhicules pour la sécurité. Les informations extraites peuvent être utilisées pour plusieurs intérêts, comme le contrôle d'accès à des zones sécurisées ou Parking privé, la surveillance des passages aux frontières et aux péages, la recherche de véhicules suspects ou encore la lutte contre la criminalité.

Dans ce travail on va réaliser un nouveau système de reconnaissance des plaques d'immatriculation basées sur K-NN le plus proche voisin.

I.2 Historique :

A l'origine, les plaques d'immatriculation ont été inventées et mises en usage réel pour les chariots, mais pas pour les voitures. Au Canada, la plaque d'immatriculation a d'abord été introduite en 1884 pour un chariot tiré par des chevaux. La première règle de la plaque d'immatriculation du monde est appliquée en France le 14 Août 1983. Suite à cette règle, toutes les plaques d'immatriculation doivent être enregistrées avec leurs propres noms, adresse et numéro d'enregistrement. En Allemagne, la police a fourni à Mr. Chubais Barthes la plaque d'immatriculation, le 14 Avril 1899, qui est la première plaque d'immatriculation enregistrée dans le monde. Cette plaque est une pièce rectangulaire.



Figure I. 1: *Exemple d'une vielle plaque d'immatriculation.*

La plaque d'immatriculation désigne une voiture d'une manière unique, et son image représente une information importante pour identifier son propriétaire. En effet, les images laques peuvent être utilisées comme support principal pour l'identification homme/véhicule, et pour la gestion des informations des plaques d'immatriculation afin de contrôler le transport public [1].

I.3 Domaine d'utilisation des systèmes de reconnaissance des plaques :

La reconnaissance des plaques d'immatriculation (Licence Plate Recognition) des véhicules est une partie critique du transport intelligent moderne, depuis que la plaque d'immatriculation est devenue un moyen simple et efficace pour identifier les véhicules.

Le système de reconnaissance des plaques a une large gamme d'applications, à savoir :

- **Gestion de parking :** Facilitation voire automatisme du paiement, des entrées et des sorties.
- **Routes à péage :** Permission d'un passage rapide via les postes de péage
- **Gestion du trafic des aéroports :** Seuls les véhicules autorisés peuvent accéder aux voies réservées aux transports publics/taxis.
- **Contrôle d'accès :** La porte s'ouvre uniquement pour les véhicules autorisés. Tous les véhicules de visite sont automatiquement enregistrés
- **Alertes véhicule :** La solution (système) émet une alerte automatique lorsqu'un véhicule figurant sur une liste de surveillance est détecté [2].

I.4 Système LPR :

Un système LPR se compose de deux éléments principaux: détection de la plaque d'immatriculation et la reconnaissance de caractères. La détection de la plaque d'immatriculation est une étape cruciale dans un système de LPR. La qualité d'un algorithme de détection influe sur la précision de la reconnaissance de la plaque d'immatriculation.

Donc, de nombreuses recherches d'identification de la voiture ont été traitées, et la plupart des algorithmes existants de détection de plaque d'immatriculation sont limités par diverses conditions contrôlées telles que les milieux fixes [3], la couleur connue [4], ou la distance connue entre la caméra et les véhicules [5].

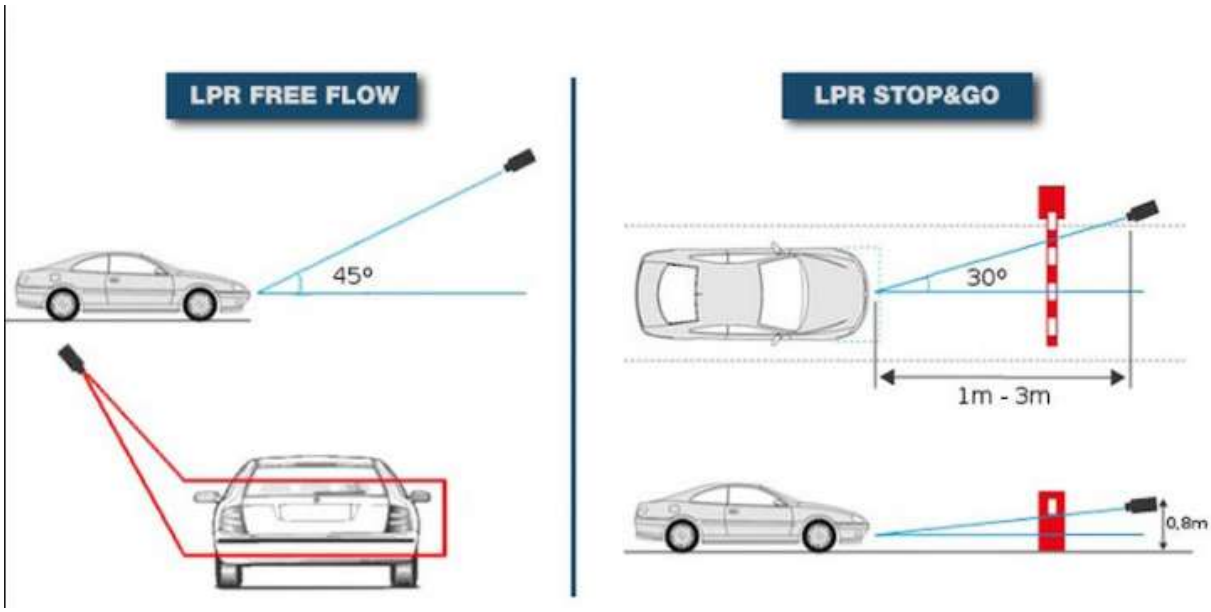


Figure I. 2:*Exemple de la position de la camera et la plaque.*

Par conséquent, le problème de la détection des plaques d'immatriculation sous des environnements complexes présente toujours un problème.

Il existe deux types d'approches pour la reconnaissance des caractères de plaque. La première approche est la reconnaissance optique des caractères (OCR) [6]. L'autre type d'approches utilise des méthodes d'apprentissage pour identifier les caractères [7].

Les deux approches ont atteint des taux de précision élevés dans des conditions contrôlées, et lorsque les caméras sont montées dans des endroits fixes sans mobilité. Toutefois, lorsque les conditions d'éclairage sont mauvaises ou lorsque les véhicules sont en mouvement, la précision est altérée.

L'étape d'extraction des caractéristiques représente une phase primordiale dans les systèmes LPR. Dans la littérature on trouve plusieurs algorithmes. Rob G. J. Wijnhoven et al

ont présenté un système de reconnaissance des plaques pour l'ouverture de la porte du garage, et la phase d'extraction des caractéristiques utilise l'histogramme des gradients orientés [8]. Dhiraj Ahuja et al. , ont proposé Un système LPR en utilisant les ondelettes et les réseaux de neurones. Pour les phases de détection et d'extraction de caractéristiques ils ont utilisé les ondelettes [9].

Dans ce travail on va réaliser un nouveau système de reconnaissance des plaques basé sur K-NN le plus proche voisin.

Dans notre projet, nous nous intéressons à la reconnaissance et la lecture automatique des plaques d'immatriculation à partir de captures d'images prises sur le devant ou à l'arrière des véhicules.

I.5 Organisation générale d'un système de reconnaissance :

Quelque soit le système de reconnaissance des caractères, il fait appel des phases suivantes [10] :

- Acquisition (scanning, Numérisation).
- Prétraitement.
- Segmentation à des caractères séparés ou segments reliés à un caractère.
- Extraction des caractéristiques.
- Classification, suivis éventuellement d'une phase de post-traitement.

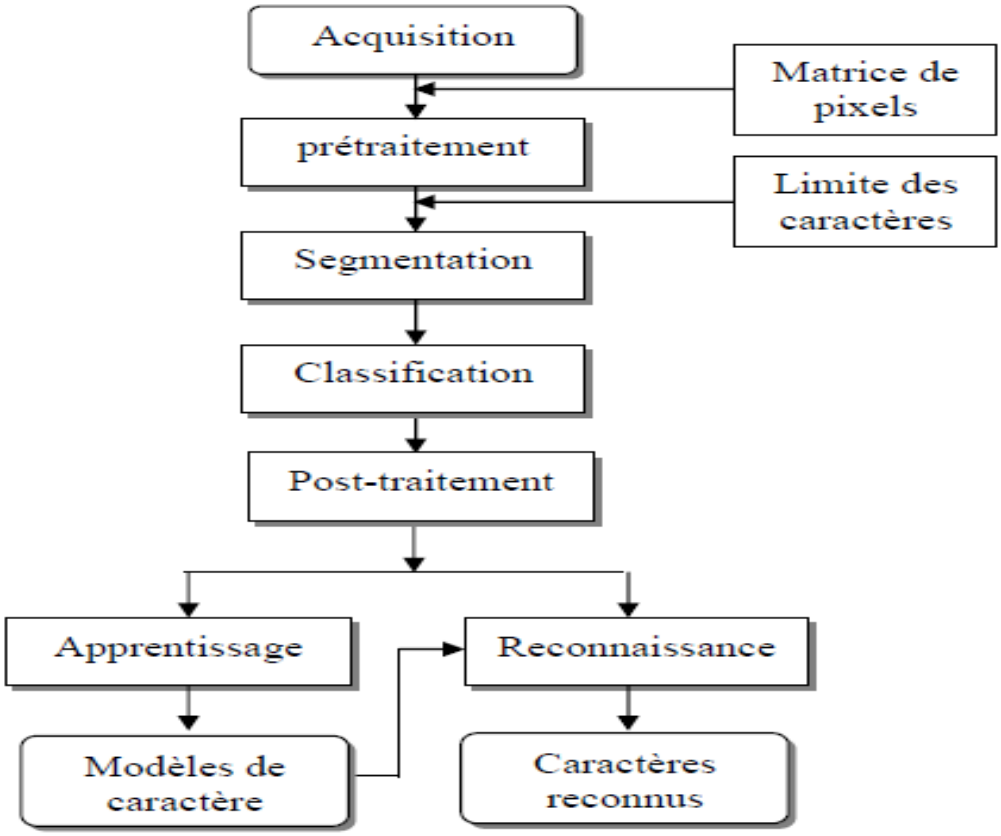


Figure I. 3: *Schéma général d'un système de reconnaissance de caractères.*

I.5.1 Phase d'acquisition :

La phase d'acquisition consiste à capter l'image d'un texte au moyen des capteurs physiques (scanner, caméra,…) et de la convertir en grandeurs numériques adaptés au système de traitement avec un minimum de dégradation possible [11]. La numérisation consiste à [12]:

- Eliminer les bruits.
- Bouchage des trous (gap filling).
- Translation d'échelle.
- Normalisation.
- Binairisation.

I.5.2 Phase de prétraitements :

Le prétraitement est le processus permettant qui couvre de la forme originale de l'image d'entrée à une forme squelettique ce qui permet de simplifier la phase d'extraction de caractéristiques [13]. Il s'agit essentiellement de réduire le bruit superposé aux données et essayer de ne garder que l'information significative de la forme représentée. Le bruit peut être

dû aux conditions d'acquisition (éclairage, mise incorrecte du document, …) ou encore à la qualité du document d'origine [14,15].

Parmi les opérations de prétraitement généralement utilisées nous pouvons citer :

- L'extraction des composantes connexes.
- Le redressement de l'écriture.
- Le lissage.
- La normalisation.
- La squelettisation.

I.5.3 Phase de segmentation :

Après que l'image soit acquise et prétraitée, nous allons maintenant extraire les différentes parties logiques de l'image en procédant d'abord par la séparation des blocs de texte et des blocs graphiques [16], ensuite on extrait les lignes de chaque bloc de texte (par une projection partielle), nous procédons à un suivi de contour partiel de chaque ligne [17], finalement de chaque ligne nous pouvons extraire les mots depuis lesquels on peut avoir les caractères (ou parties du caractère) [18].

I.5.4 Phase de classification :

Le processus de classification consiste à déterminer à quelle classe appartient une entrée de données (voir figure I.3) [19]. La classification dans un système OCR regroupe deux tâches : l'apprentissage, la reconnaissance et la décision.

A cette étape les caractéristiques de l'étape précédente sont utilisées pour identifier un segment de texte et l'attribuer à un modèle de référence [20].

I.5.5 Etape d'apprentissage :

Avant de pouvoir prendre une décision, il faut acquérir des connaissances et les organiser en modèles de référence ou classes. C'est durant la phase d'apprentissage qu'est réalisé ce travail. Lorsque le concepteur (ou professeur) indique le nom de la forme d'entrée, l'apprentissage est dit supervisé. Si la construction des classes est automatique, on parle d'apprentissage non supervisé [21].

I.6 Systèmes de reconnaissance de caractères OCR :

I.6.1 Historique d'O.C.R :

La reconnaissance de l'écriture est mieux connue sous le nom d'O.C.R (Optical Character Recognition). Les premières tentatives remontent aux années 1900 par TYURIN, au cours desquelles on inventa le scanner à balayage pour la télévision ; et en l'an 1912 par ALBE et en l'an 1925 par Thomas, qui ont mimé l'interprétation Humaine de l'informatique visuelle , un point de transformation est apparue avec l'invention du premier ordinateur en 1946 par MAUCHLY et ECKERT , quelques années plus tard les premières expériences en reconnaissance de caractères ont pu être réalisées ;pendant les année soixante, et soixante dix, les premiers systèmes de l'écriture automatique du texte imprimé ont vu le jour [22].

Aussi, en 1975 les japonais utilisaient couramment les lecteurs qui déchiffrent le code postal inscrit à la main ou tapé à la machine. Dans la même période, le Français CONTER construisait un système de lecture automatique de texte imprimé destiné aux non voyants, par la suite la compagnie américaine KURZWELL a amélioré le système précédent en proposant les machines à lire pour aveugle formulant le texte à haute voix par synthèse vocale [24].

Durant cette phase les chercheurs se sont heurtés à de nombreuses difficultés outre la complexité du problème de la reconnaissance due à la grande variabilité de l'écriture manuscrite, La non disponibilité de mémoire et de puissance de calcul pour la réalisation des systèmes concrets opérant en temps réel, par contre depuis 1980 où les récents progrès électroniques et plus particulièrement l'avènement de calculateurs puissants a faible coût ont permis de résoudre ce type de problème et les recherches en reconnaissance manuscrite se sont multipliées de manière spectaculaire, et de nombreuses nouvelles techniques ont vu le jour [25].

I.6.2 Introduction :

Toute information écrite peut être reprise dans une chaine de traitement informatisée à différentes fins : la rédaction et l'édition de rapports, la diffusion de documents dans un système de messagerie . . . conduisent à exploiter des informations disponibles seulement sur papier.

La reconnaissance optique des caractères (OCR) est une opération informatique rapide permettant de réaliser la transformation d'un texte écrit sur papier en un texte sous forme d'un fichier informatique en représentation symbolique (par exemple pour les écritures latines, le codage opéré est le code ASCII (American Standard code for information interchange),

tandis que pour l'arabe on utilise généralement le code ASMO (Arabic Standard Métrologie Organisation).

I.6.3 Définition :

La reconnaissance optique de caractères est un procédé permettant de récupérer les Symboles dans les images de textes numérisées. Les images consistent en des matrices de pixels. La tâche d'un OCR consiste à segmenter les images , mots, caractères puis effectuer la reconnaissance des caractères. Il résulte une transcription textuelle de l'image par laquelle des traitements automatiques sont possibles : recherche de mots, de noms, résumé.

En général, l'OCR signifie le décodage de n'importe quel chiffre imprimè ou information symbolique, par exemple les formules mathématiques, plaque d'immatriculation…etc [26].

I.6.4 Domaine d'application O.C.R :

L'O.C.R connaît plusieurs applications pratiques dans plusieurs domaines parmi lesquels on cite :

- **La lecture des tickets de transport aérien :** Chaque place réservée nécessite trois enregistrements : un auprès de la compagnie, un auprès de l'agence de voyage et un pour le voyageur. Afin de limiter le grand nombre de billets ainsi créés et d'éviter l'attente avant embarquement, de nombreuses compagnies ont recours à un système d'identification automatique qui lit le ticket et compare les indications avec la base de données de chaque vol.

- **La lecture de formulaires** : De nombreuses enquêtes ou aches de renseignements utilisent des formulaires préimprimées. L'utilisation d'un système de reconnaissance, capable de lire directement les données dans les zones réservées permet d'effectuer rapidement la saisie de ces documents [27].

- **La lecture des passeports** : Certaines douanes sont équipées de lecture de passeports afin d'identifier chaque voyageur. Le système permet de lire le nom, la nationalité, le numéro de passeport et aussi de contrôler directement auprès des bases de données des services d'immigration, l'autorisation de séjour.

- **Domaine bancaire :** Pour l'authentification des chèques par les banques.
- **Assistance à l'éducation :** Pour la reconnaissance et la traduction de textes en braille et l'apprentissage de l'écriture (lecture, écriture) des photo-senseurs, Et des simulateurs tactiques sont utilisés.
- **Lecture des adresses postales :** la lecture des codes postaux manuscrits associés à la lecture des noms de villes a permis d'atteindre le développement des machines de tri automatique de courrier (chiffres et objets plats).
- **La police et la sécurité :** pour la reconnaissance des numéros minéralogiques pour le contrôle routier, l'authentification de manuscrits et l'identité des scripteurs.
- On utilise autrefois le terme O.C.R ou reconnaissance de caractère qui appartient au domaine lecture automatique de document qui couvre trois axes :
  - ✓ Lecture optique des caractères imprimés.
  - ✓ La reconnaissance de caractères manuscrits.
  - ✓ La reconnaissance de l'écriture manuscrite cursive [28].

I.6.5 Différents aspects de l'OCR :

Il est très difficile de créer un système OCR capable de reconnaître n'importe quel écriture ou fonte. Tout dépend de l'application vissée ou voulu et des données à traiter. Généralement, les systèmes OCR sont classés en se basant sur trois critères [29]:

- **Outil d'acquisition**: Les systèmes qualifiés de « en-ligne » ou « hors-ligne »,
- **Approches de reconnaissance:** approches globales ou analytiques selon que l'analyse s'opère sur la totalité du mot, ou par segmentation en caractères.
- **Nature des traits caractéristiques:** approches statistiques, structurelles ou stochastiques.

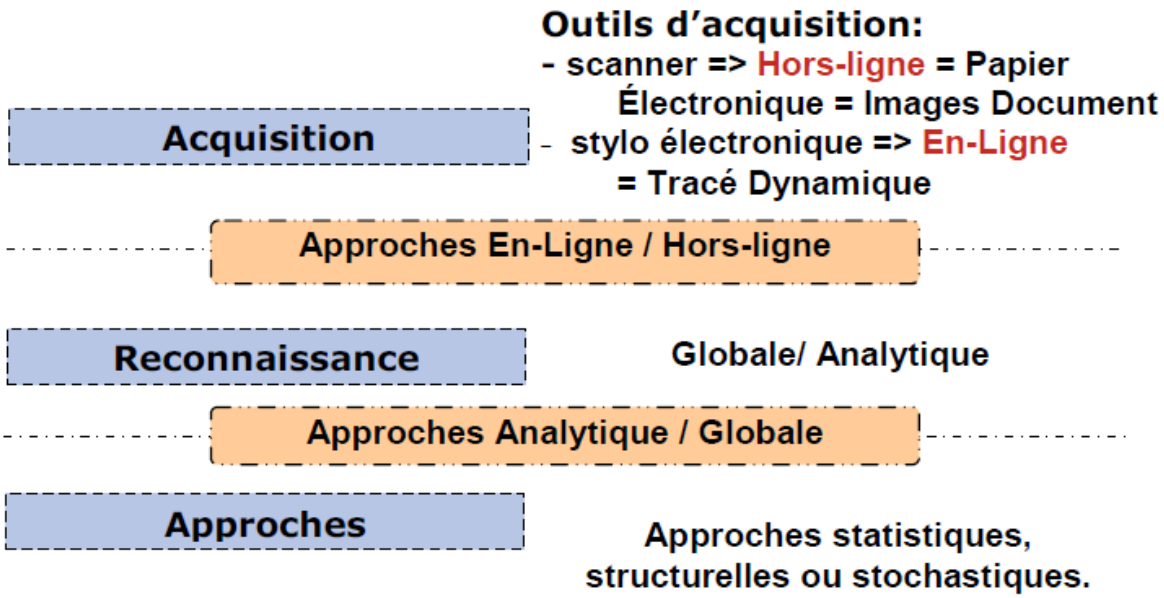


Figure I. 4: *Différents aspects de l'OCR.*

I.6.6 Classification de l'OCR :

Il n'existe pas de système universel d'OCR qui permet de reconnaître n'importe quel caractère dans n'importe quelle fonte. Tout dépend du type de données traitées et bien évidemment de l'application visée [9].

Il existe plusieurs modes de classification des systèmes OCR parmi lesquels on peut citer [7] :

- Les systèmes << en-ligne >> ou << hors-ligne >> suivant le mode d'acquisition.
- Les approches globales ou analytiques selon que l'analyse s'opère sur la totalité du mot, ou par segmentation en caractères.

I.6.6.1 Reconnaissance en-ligne et hors-ligne :

Ce sont deux modes différents d'OCR, ayant chacun ses outils propres d'acquisition et ses algorithmes correspondants de reconnaissance [30].

1) **La reconnaissance en-ligne (on-line) :** Ce mode de reconnaissance s'opère en temps réel (pendant l'écriture). Les symboles sont reconnus au fur et à mesure qu'ils sont écrits à la main. Ce mode est réservé généralement à l'écriture manuscrite. c'est une approche où la reconnaissance est effectuée sur des données à une dimension. L'écriture est représentée comme un ensemble de points dont les coordonnées sont fonction du temps [31]. La reconnaissance en-ligne présente un avantage majeur c'est la possibilité de correction et de modification de l'écriture de manière interactive vu la

réponse en continu du système [32]. L'acquisition de l'écrit est généralement assurée par une tablette graphique munie d'un stylo électronique.



Figure I. 5:*Exemple de reconnaissance en ligne.*

2) **La reconnaissance hors-ligne (off-line) :** Démarre après l'acquisition. Elle convient aux documents imprimés et les manuscrits déjà rédigés. Ce mode peut être considéré comme le cas le plus général de la reconnaissance de l'écriture. Il se rapproche du mode de la reconnaissance visuelle. L'interprétation de l'information est indépendante de la source de génération [33].

La reconnaissance hors-ligne peut être classée en plusieurs types :

- ✓ Reconnaissance de texte ou analyse de documents.
- ✓ Reconnaissance de l'imprimé ou du manuscrit.

I.6.7 Problèmes liés à l'OCR :

Divers problèmes compliquent le processus de reconnaissance, ce qui rend la tâche de l'OCR très délicates, parmi ces problèmes nous pouvons citer [34]:

- ❖ Disposition spatiale du texte.
- ❖ Nombre de scripteurs.
- ❖ Taille du vocabulaire.

Conclusion :

Dans ce chapitre, nous avons essayé de donner une vue générale sur la reconnaissance des caractères et les étapes suivis pour la réaliser passant par quelques concepts de bases de l'OCR et les principaux problèmes rencontrés dans ce domaine. En plus les principales méthodes de reconnaissance en général.

Ensuite nous avons abordé les différentes étapes intervenant dans la conception d'un système de reconnaissance de caractères.

CHAPITRE II

APPROCHES DE CLASSIFICATION

II.1 Introduction :

La classification des images est l'une des plus importantes branches de l'intelligence artificielle; son application semble être dans une direction prometteuse dans le développement de la reconnaissance des caractères en reconnaissance optique de caractères (OCR). Ce dernier a fait l'objet de nombreuses études au cours des cinquante dernières années et progressé au niveau suffisant pour produire des technologies applications. La puissance de calcul en croissance rapide permet la mise en œuvre des méthodologies actuelles de CR et crée également une demande croissante pour de nombreuses applications émergentes domaines, qui nécessitent des méthodologies plus avancées. Chercheurs pour la reconnaissance des langages indiens et des scripts sont comparativement moins avec d'autres langues. Il y a beaucoup de différents algorithmes d'apprentissage automatique utilisés pour l'image classement de nos jours. Dans cet article, nous discutons des caractéristiques de certaines méthodes de classification telles que le support Machines à vecteurs (SVM) et K "efronthood (K- NN)" qui ont été appliqués aux caractères oriya. Nous allons discuter de la performance de chaque algorithme pour la classification des caractères.

Pour notre étude, nous allons nous baser sur des méthodes d'apprentissage supervisées, à savoir les KNN pour la reconnaissance des plaques d'immatriculations. Dans la section suivante on présente une brève description des méthodes de classification.

II.2 Approche de classification :

II.2.1 Introduction :

La classification des images est l'une des plus importantes branches de l'intelligence artificielle; son application semble être dans une direction prometteuse dans le développement de la reconnaissance des caractères en reconnaissance optique de caractères (OCR). Reconnaissance de caractère a fait l'objet de nombreuses études au cours des cinquante dernières années et progressé au niveau suffisant pour produire des technologies applications. La puissance de calcul en croissance rapide permet la mise en œuvre des méthodologies actuelles de CR et crée également une demande croissante pour de nombreuses applications émergentes domaines, qui nécessitent des méthodologies plus avancées. Chercheurs pour la reconnaissance des langages indiens et des scripts sont comparativement moins avec d'autres langues. Il y a beaucoup de différents algorithmes d'apprentissage automatique utilisés pour l'image classement de nos jours. Dans cet article, nous discutons des caractéristiques de

certaines méthodes de classification telles que le support Machines à vecteurs (SVM) et K " efronthood (K-NN) qui ont été appliqués aux caractères oriya. Nous allons discuter des performances de chaque algorithme pour la classification des caractères.

Pour notre étude, nous allons nous baser sur des méthodes d'apprentissage supervisées, à savoir les KNN pour la reconnaissance des plaques d'immatriculations. Dans la section suivante on présente une brève description des méthodes de classification [35].

II.2.2 Définition de la classification :

La classification est l'une des techniques les plus anciennes d'analyse et de traitement de données, Une classe est un ensemble d'éléments qui sont semblables entre eux et qui sont dissemblables à ceux d'autres classes. La classification repose sur des objets à classer. Les objets sont localisés dans un espace de variables (ont dit aussi attributs, caractéristiques ou critères). Il s'agit de les localiser dans un espace de classes. Ce problème n'a de sens que si on pose l'existence d'une correspondance entre ces deux espaces. Résoudre un problème de classification, c'est trouver une application de l'ensemble des objets à classer, décrits par les variables descriptives choisies, dans l'ensemble des classes. L'algorithme ou la procédure qui réalise cette application est appelé classifieur [36].

On retrouve trois approches de classification :

II.2.2.1 Classification non supervisé : Est une méthode d'apprentissage automatique [11].

Il s'agit pour un logiciel de diviser un groupe hétérogène de données, en sous-groupes de manière que les données considérées comme les plus similaires soient associées au sein d'un groupe homogène et qu'au contraire les données considérées comme différentes se retrouvent dans d'autres groupes distincts l'objectif étant de permettre une extraction de connaissance organisée à partir de ces données.

La recherche d'une partition des données revient à regrouper celles-ci selon une certaine mesure de similarité ou de dis similarité Contrairement à l'apprentissage supervisé, dans l'apprentissage non-supervisé il n'y a pas de signe qui explicite les étiquettes [37,38].

II.2.2.2 Classification Semi supervisé :

L'apprentissage semi-supervisé est une classe de techniques d'apprentissage qui utilise un ensemble de données étiquetées et non-étiquetés. Il se situe ainsi entre l'apprentissage supervisé qui n'utilise que des données étiquetées et l'apprentissage non-supervisé qui n'utilise que des données non-étiquetées. Il a été démontré que l'utilisation de données non-étiquetées,

en combinaison avec des données étiquetées, permet d'améliorer significativement la qualité de l'apprentissage [39].

II.2.2.3 Classification Supervisée : (Un ensemble de points étiquetés) c'est une technique d'apprentissage automatique où l'on cherche à produire automatiquement des règles à partir d'une base de données d'apprentissage. (En général des cas déjà traités et validés).

Dans le cas supervisé, les classes d'appartenance des données sont connues [40]. La recherche des frontières entre les classes peut être effectuée par la recherche des fonctions discriminantes. La classification supervisée ou la classification inductive a pour objectif de chercher à expliquer et à prédire l'appartenance de documents à des données connues a priori [41]. Dans le cadre de ce projet de fin d'étude, nous entamerons l'apprentissage supervisé pour des problèmes de classification.

Dans la suite, nous présenterons les principaux algorithmes de classification supervisée proposés dans la littérature. Il ne s'agit pas de faire une présentation exhaustive de toutes les méthodes mais seulement de préciser les méthodes les plus classiques que nous utiliserons dans le cadre de notre travail en fonction de leurs propriétés particulières.

II.3 Types des méthodes de classification supervisée :

Voyons maintenant le premier type des méthodes de classification supervisée qui est le réseau de neurone :

II.3.1 Réseaux de neurones : Perceptron multicouche :

Le perceptron multicouche(PMC) est l'un des réseaux de neurones les plus utilisés actuellement, pour la classification supervisée notamment. Dans un réseau de neurones, deux modes de fonctionnement peuvent être distingué. Dans le premier, les paramètres du réseau sont ajustés grâce à la présentation d'exemples pour lesquels on connaît la réponse désirée : ce mode de fonctionnement est appelé « phase d'apprentissage ». Dans le deuxième mode, il s'agit d'exploiter le réseau en lui présentant des données inconnues : ce mode d'utilisation est souvent appelé « phase de reconnaissance ». Le perceptron est organisé en plusieurs couches. La première est reliée aux entrées, puis ensuite chaque couche reliée à la couche précédente. C'est la dernière couche qui produit les sorties du PMC [42]. De telle sorte que les neurones sont reliés entre eux par des connexions pondérées, ce sont les poids de ces connexions qui gouvernent le fonctionnement du réseau :

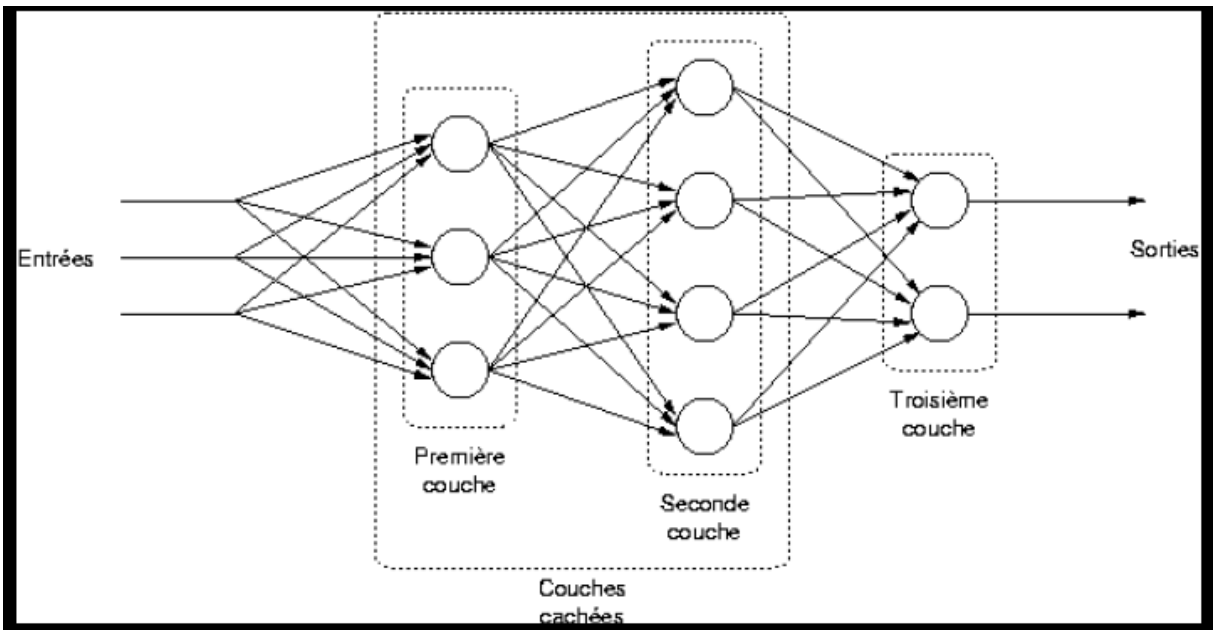


Figure II. 1 : *Schéma illustrant un perceptron multicouche avec deux couches cachées.*

Pour le premier mode de fonctionnement de perceptron multicouche on distingue trois algorithmes :

- Algorithme de rétro-propagation du gradient.
- Algorithme de gradient conjugué.
- Méthodes de second ordre.

Et l'article [42] décrit la méthode exacte du fonctionnement d'adaptation des poids du perceptron multicouche.

a) les avantages de ces réseaux de neurones [44] :

- Les réseaux de neurones sont capables de tolérer des données bruitées, ainsi que capable de classer les modèles sur lesquels ils ne sont pas été formés.
- Ils peuvent être utilisés lorsque nous avons peu d'informations sur la relation entre les attributs et les classes.
- Bien adapté pour les entrées et les sorties d'une valeur continue.
- Le succès sur plusieurs applications du monde réel comme la reconnaissance manuscrite de caractère, de la pathologie et de médecine de laboratoire, et beaucoup plus.

b) Parmi ses inconvénients on cite [45]:

- Nécessitent un long temps d'apprentissage, donc ils sont adaptable aux applications dont ceci est faisable.
- Pauvre interprétation de la connaissance représentée sous la forme d'un réseau d'unités connectées par des liens pondérés.
- Exigent un nombre de paramètres qui doivent être déterminées empiriquement, par exemple la topologie du réseau et sa structure, le nombre de couches cachées, le nombre d'unités dans chaque couche cachée et dans la couche de sortie.

Dans la section suivante on va entamer le deuxième type des méthodes de classification supervisée qui représente les K plus proches voisins :

II.3.2 K-plus proches voisins :

Les K plus proches voisins, connus en anglais sous le nom K-Nearest Neighbors (K-NN) L'algorithme k-plus proches voisins est l'un des algorithmes les plus simples utilisés dans la classification supervisée et le plus connu dans les algorithmes d'instance. Cet algorithme ne nécessite aucune phase de formation. Ceci est que l'ensemble d'entraînement associé à une fonction de distance et d'une fonction de sélection de la classe en fonction des classes des plus proches voisins, qui constitue le modèle.

Cet Algorithme a prouvé son efficacité face au traitement de données textuelles (Yang, 1997). La phase d'apprentissage consiste à stocker les exemples étiquetés. Le classement de nouvelles entrées s'opère en calculant la distance entre la représentation vectorielle de la donnée et celle des exemples du corpus; les k éléments les plus proches sont sélectionnés et l'entrée à classifier, est assigné à la classe majoritaire [46].

II.3.2.1 Comment fonctionne l'algorithme KNN?

En KNN, K est le nombre de voisins les plus proches. Le nombre de voisins est le facteur décisif. K est généralement un nombre impair si le nombre de classes est 2. Lorsque K=1, l'algorithme est appelé algorithme du plus proche voisin. C'est le cas le plus simple. Supposons que P1 soit le point pour lequel l'étiquette doit prédire. Tout d'abord, vous trouvez le point le plus proche de P1, puis l'étiquette du point le plus proche attribué à P1.

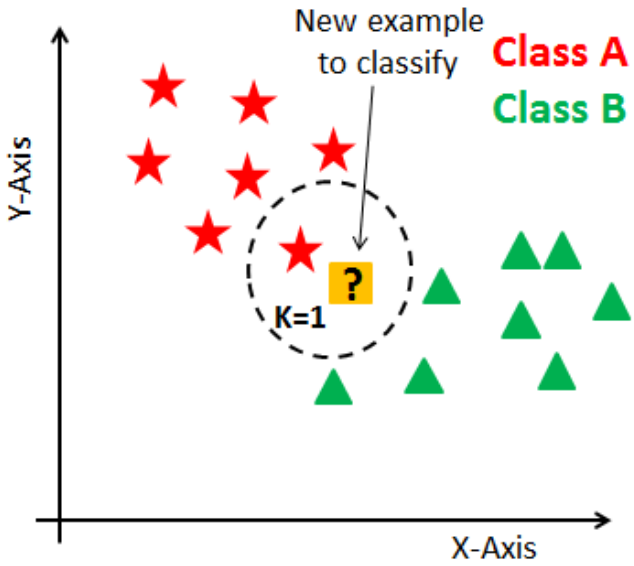


Figure II. 2:*exemple de classification lorsque K=1.*

Supposons que P1 soit le point pour lequel l'étiquette doit prédire. Tout d'abord, vous trouvez le k point le plus proche de P1, puis vous classifiez les points par vote majoritaire de ses k voisins. Chaque objet vote pour sa classe et la classe avec le plus de votes est prise comme prédiction. Pour rechercher des points similaires les plus proches, vous devez calculer la distance entre les points à l'aide de mesures de distance telles que la distance euclidienne, la distance de Hamming, la distance de Manhattan et la distance de Minkowski. KNN a les étapes de base suivantes:

- Calculer la distance.
- Trouver les plus proches voisins.
- Votez pour les étiquettes.

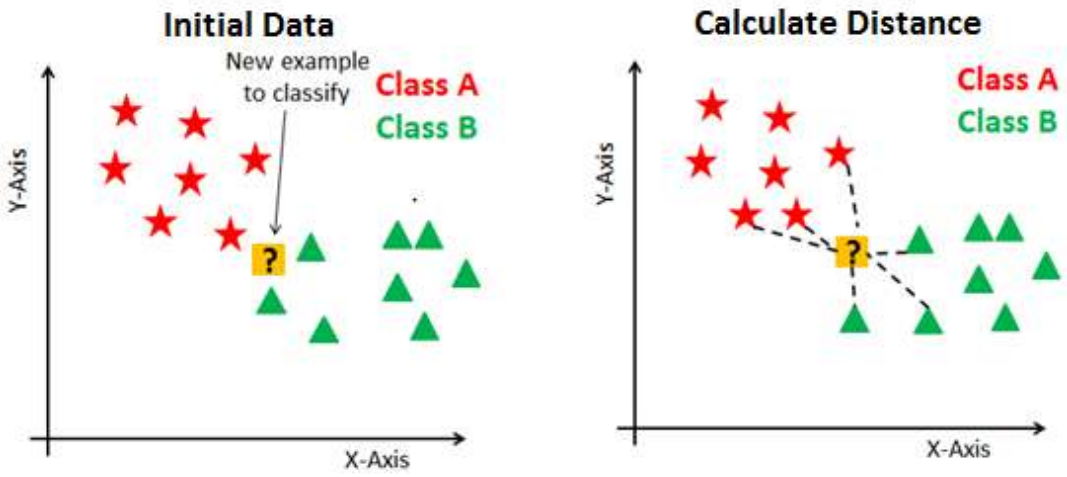


Figure II. 3:*Les étapes de base de KNN.*

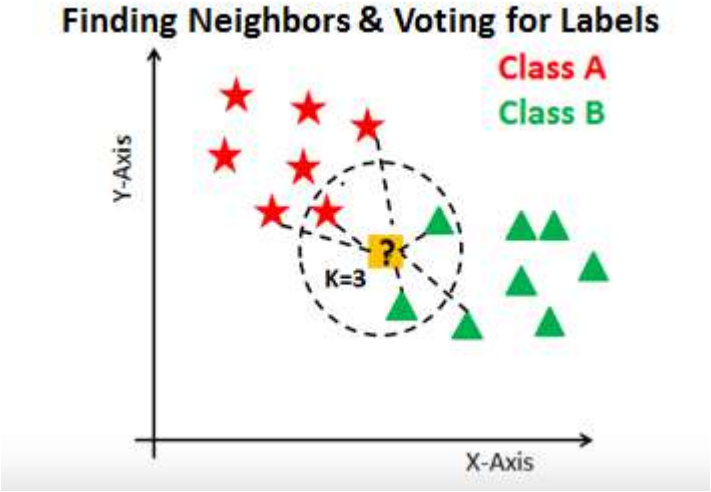


Figure II. 4: *3eme étape :trouver les voisins.*

II.3.2.2 Exemple de k-NN :

Pour prédire la classe d'un exemple donné, l'algorithme cherche les K plus proches voisins de ce nouveau cas et prédit la réponse la plus fréquente de ces K plus proches voisins [47].

Le principe de décision consiste tout simplement donc a calcules la distance de l'exemple inconnu a tous les échantillons fournis. L'exemple est alors affecté à la classe majoritaire représenté parmi ces K échantillons. La méthode utilise deux paramètres : le nombre K et la fonction de similarité pour comparer le nouvel exemple aux exemples déjà classés [48].

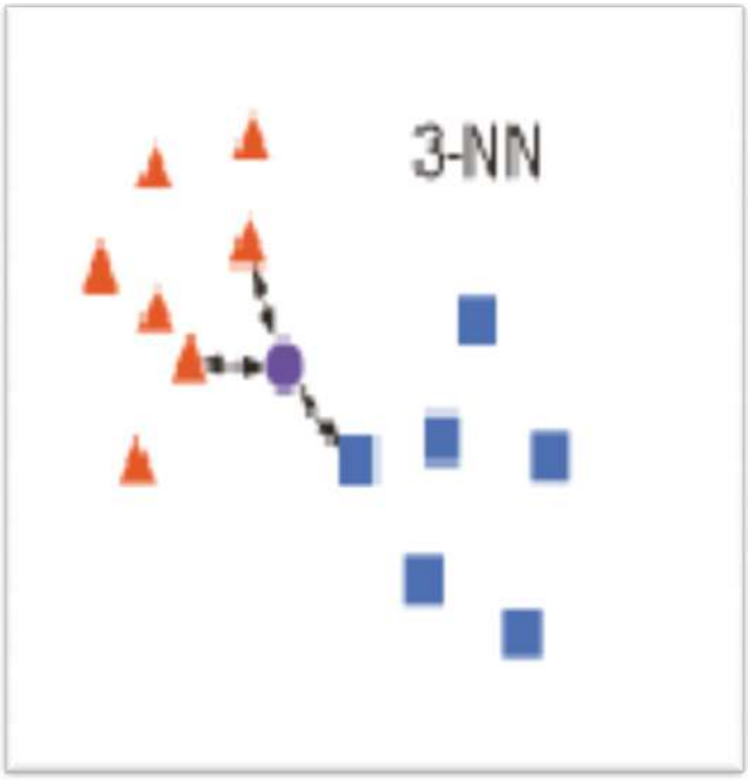


Figure II. 5: *Exemple de classification par KNN.*

II.3.2.3 L'extension de 1-NN à k-NN se fait comme suit :

a) Trouver les k plus proches observations.

b) Utiliser une règle de décision à la majorité pour classer une nouvelle observation :

● L'avantage est que de grandes valeurs de k produisent un lissage qui réduit le risque d'apprentissage du bruit dans les données d'apprentissage.

● Typiquement les valeurs de k sont choisies dans des échelles de quelques unités à quelques dizaines plutôt que quelques milliers.

● Si on choisit k = n (nombre d'observations dans l'ensemble d'apprentissage) la classe retenue sera celle qui a la majorité dans les données d'apprentissage, indépendamment de l'observation inconnue (u 1, u 2,..., u p) [49].

➢ **Le choix de k :** Le paramètre k doit être déterminé par l'utilisateur : k€N, il est utile de choisir k impair pour éviter les votes égalitaires. Le meilleur choix de k dépend du jeu de donnée. La fixation du paramètre k est délicate, une valeur très faible va

engendrer une forte sensibilité. Un K trop grand va engendrer un phénomène d'uniformisation des décisions.

Pour remédier à ce problème, il faut tester plusieurs valeurs de k et choisir le k optimal qui donne un meilleur taux de classification [50].

➢ **Sur la prédiction :** L'idée des k-NN peut facilement être étendue à la prédiction de valeurs continues (comme dans le cas de la régression linéaire multiple) en faisant une prédiction sur la valeur moyenne de la variable dépendante sur les k plus proches voisins. Souvent cette moyenne est pondérée par un poids décroissant en fonction de la distance croissante à partir du point à partir duquel la prédiction est requise [51].

➢ **Défauts des k-NN :** Bien qu'il n'y ait pas besoin de temps pour estimer les paramètres, le temps pour calculer les k voisins peut être prohibitif. Un certain nombre d'idée ont été imaginées pour palier ce problème :

- ✓ diminuer le coût en calcul de la distance en travaillant sur un espace de dimension plus petit (en utilisant une méthode de réduction de dimension du type analyse en composante principale).
- ✓ utiliser une technique de représentation sophistiquée de type arbre de recherche pour accélérer le processus d'identification d'un voisin. Ces méthodes permettent de tomber sur un « presque voisin » pour aller plus vite.
- ✓ Editer les données d'apprentissage pour éliminer les observations redondantes. Parfois autour d'une observation on trouve des observations ressemblantes qui appartiennent à la même classe[52].

II.3.2.4 Les avantages de la méthode des K plus proches voisins on cite [53] :

- ➢ Facile à comprendre et facile à mettre en œuvre.
- ➢ Plus rapide dans l'entraînement que les méthodes d'apprentissage avides.
- ➢ Performant pour les applications dont on trouve qu'un exemple peut avoir plusieurs étiquettes de classe.

II.3.2.5 Les inconvénients de la méthode des K plus proches voisins on cite [55] :

- ➢ Ils sont Sensibles à la structure locale des données.
- ➢ Ils ont de grands besoins de stockage.

- Ils sont sensibles à la sélection de la fonction de similarité qui est utilisée pour comparer des instances.

II.3.2.6 Reconnaissance de caractères en utilisant KNN :

L'objectif est de développer une méthode efficace qui utilise une image personnalisée pour former le classificateur. Cet extrait d'OCR caractéristiques distinctes de l'image d'entrée pour la classification de son contenu comme des caractères spécifiquement des lettres et des chiffres. L'entrée dans le système est images numériques contenant les motifs à classer.

L'analyse et la reconnaissance des motifs dans les images sont en train de devenir plus complexe, mais facile avec les progrès technologiques connaissance. Par conséquent, il est proposé de développer des technologies sophistiquées stratégies d'analyse des modèles pour faire face à ces difficultés. Le travail actuel implique l'application de la reconnaissance de formes en utilisant KNN pour reconnaître le texte manuscrit ou imprimé [56].

II.3.2.6.1 Les étapes algorithmiques sont les suivantes:

- ✓ L'image d'entraînement ayant le jeu de caractères dans un format différent est lu et traité. Image de formation est converti en échelle de gris, floue, seuil et image aplatie de sorte qu'il est plus facile pour le système de comprendre les caractéristiques de l'image et différencier entre les objets (personnages) et indésirables Contexte.
- ✓ À partir des contours avec données, les entités sont extraites pour chaque caractère et stocké dans un tableau numpy. Ensuite ceci tableau et un tableau contenant des étiquettes sont combinés et stocké dans un fichier texte.
- ✓ Le deuxième module est destiné à la formation et aux tests. Le les fichiers texte sauvegardés sont chargés. Puis un classificateur KNN L'objet est créé à l'aide de CV2, qui est formé à l'aide du fichier texte.
- ✓ Maintenant, l'image à tester est lue et encore l'image est converti et traité pour extraire les caractéristiques des contours avec des données.
- ✓ Ensuite, les contours avec des données valides sont vérifiés et séparément stocké dans une liste.
- ✓ Ces contours sont marqués par des rectangles verts sur l'image de test et montré.
- ✓ Ces rectangles marqués en vert sont recadrés, redimensionnés seuil et aplati.

- ✓ Puis, à l'aide de la fonction de recherche la plus proche, KNN, étiquette de caractère avec la plupart des fonctionnalités correspondantes sont obtenu et affiché sous forme de chaîne.

Après avoir converti une image en image en niveaux de gris, il est analysé puis la différence d'espacement et d'espaces de pixels déterminer le texte et ses limites afin qu'il puisse analyser les caractères séparément sans aucun obstacle à l'adaptabilité. Cela a été trouvé dans cette reconnaissance.

- ✓ L'algorithme KNN est utilisé en raison de sa plus grande précision sur les problèmes multiclass non-linéaires [57].

Voyons maintenant le troisième type des méthodes de classification supervisée, qui est le SVM :

II.3.3 Support vecteur machine (SVM):

II.3.3.1 Introduction :

Parmi les méthodes à noyaux, inspirées de la théorie statistique de l'apprentissage de Vladimir Vapnik, les Machines à Vecteurs de Support (SVM) constituent la forme la plus connue. SVM est une méthode de classification binaire par apprentissage supervisé, elle fut introduite par Vapnik en 1995. Cette méthode est donc une alternative récente pour la classification.

Elle repose sur l'existence d'un classificateur linéaire dans un espace approprié. Puisque c'est un problème de classification à deux classes, cette méthode fait appel à un jeu de données d'apprentissage pour apprendre les paramètres du modèle. Elle est basée sur l'utilisation de fonctions dites noyau (kernel) qui permettent une séparation optimale des données. Dans la présentation des principes de fonctionnements, nous schématiserons les données par des « points » dans un plan [58].

II.3.3.2 1. Pourquoi les Machines à Vecteurs de Support (SVM) ?

L'algorithme des machines à vecteurs de support a été développé dans les années 90 par le russe Vladimir Vapnik. Initialement, les SVM ont été développé comme un algorithme de classification binaire supervisée. Il s'avère particulièrement efficace de par le fait qu'il peut traiter des problèmes mettant en jeu de grands nombres de descripteurs, qu'il assure une solution unique (pas de problèmes de minimum local comme pour les réseaux de neurones) et il a fourni de bons résultats sur des problèmes réels [59].

L'algorithme sous sa forme initiale revient à chercher une frontière de décision linéaire entre deux classes, mais ce modèle peut considérablement être enrichi en se projetant dans un autre espace permettant d'augmenter la séparabilité des données. On peut alors appliquer le même algorithme dans ce nouvel espace, ce qui se traduit par une frontière de décision non linéaire dans l'espace initial [60].

II.3.3.3 Définition :

Les machines à vecteurs de support (Support Vector Machine, SVM) appelés aussi séparateurs à vaste marge sont des techniques d'apprentissage supervisées destinées à résoudre des problèmes de classification. Les machines à vecteurs supports exploitent les concepts relatifs à la théorie de l'apprentissage statistique et à la théorie des bornes de Vapnik et Chervonenkis 1995. La méthode cherche alors l'hyperplan qui sépare les exemples positifs des exemples négatifs, en garantissant que la marge entre le plus proche des positifs et des négatifs soit maximale. Cela garantit une généralisation du principe car de nouveaux exemples pourront ne pas être trop similaires à ceux utilisés pour trouver l'hyperplan mais être situés d'un côté ou l'autre de la frontière. L'intérêt de cette méthode est la sélection de vecteurs supports qui représentent les vecteurs discriminant grâce auxquels est déterminé l'hyperplan. Les exemples utilisés lors de la recherche de l'hyperplan ne sont alors plus utiles et seuls ces vecteurs supports sont utilisés pour classer un nouveau cas, ce qui peut être considéré comme un avantage pour cette méthode [61].

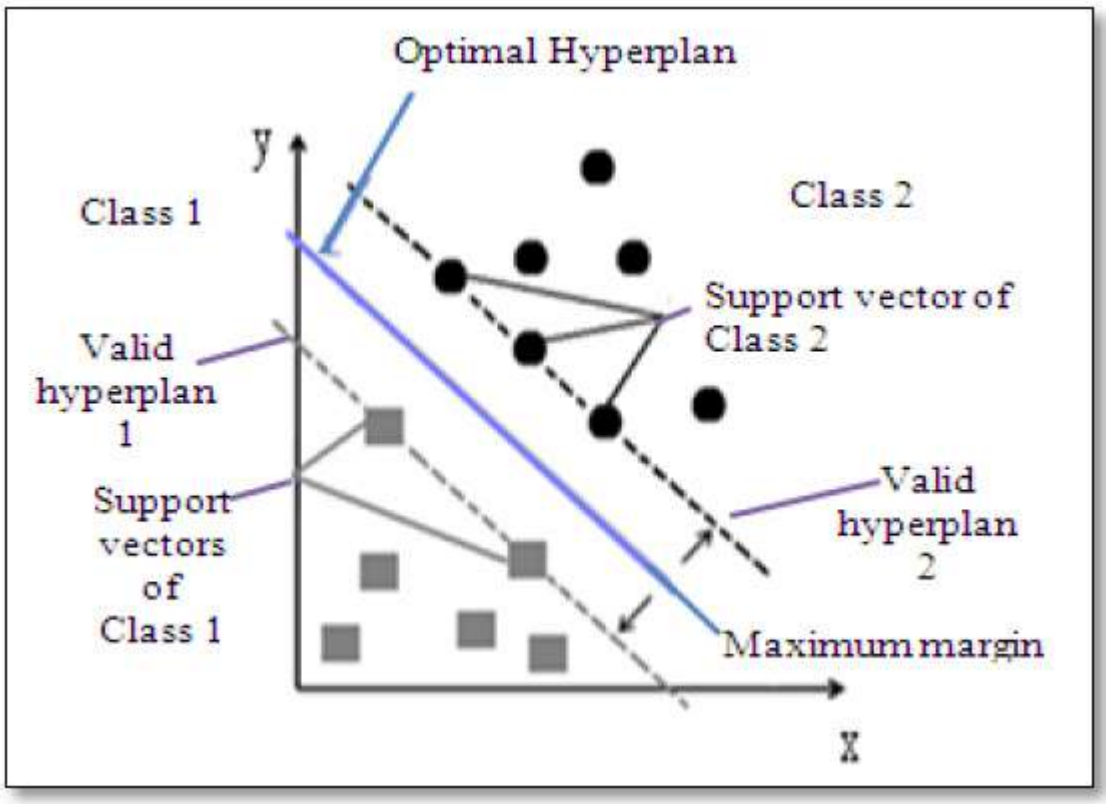


Figure II. 6 : *Illustration du principe des SVMs.*

II.3.3.4 Les avantages de la méthode SVM :

- Les SVMs possèdent des fondements mathématiques solides.
- Les exemples de test sont comparés juste avec les supports vecteur et non pas avec tous les exemples d'apprentissage.
- Décision rapide. La classification d'un nouvel exemple consiste à voir le signe de la fonction de décision f(x) [62].

II.3.3.5 Les inconvénients de la méthode SVM :

Grande quantité d'exemples en entrées implique un calcul matriciel important.

- Temps de calcul élevé lors d'une régularisation des paramètres de la fonction noyau.

Après cette brève description des méthodes non-supervisées utilisées dans ce système, la section qui suit représente l'ensemble des résultats obtenues de reconnaissance des plaques d'immatriculation [62].

Dans la section suivante on va entamer le quatrième type des méthodes de classification supervisée le HMM :

II.3.4 Modèles de Markov caches (HMM) :

II.3.4.1 Introduction :

Un modèle de Markov caché est un processus doublement stochastique, avec un processus stochastique sous-jacent non observable (d'où le mot caché), mais peut être observé à travers

un autre processus stochastique qui produit la séquence de observations Le processus caché consiste en un ensemble d'états connectés les uns aux autres par des transitions avec probabilités, alors que le processus observé consiste en un ensemble de sorties ou observations, chacune pouvant être émise par chaque état selon une certaine densité de probabilité de sortie fonction ils a plusieurs types de HMM peuvent être distingués [64].

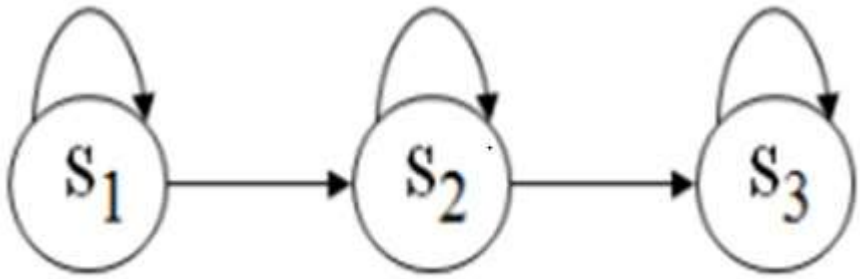


Figure II. 7: *structure of hidden states.*

II.3.4.2 Les trois problèmes des HMMs :

Comme en vient de le voir les HMM permettent de modéliser des séquence d'observation discrètes ou continus il permettent de résoudre principalement trois grand problème :

- **L'apprentissage** : étant sonné un ensemble de séquence déterminer les paramètres d'un model de markov caché d'architecture fixé pour maximiser les probabilités démission de séquence
- **L'évaluation dune séquence** étant donné un modèle de markov cache et une séquence déterminer quelle est la probabilité démission de cette séquence suivant ce modèle
- **La rechercher su chemin le plus probable :** étant donné un modèle de markov caché et une séquence déterminer la suit d'état qui la probabilité d'observation de cette séquence [65].

II.3.4.3 Les avantage des HMMs [66] :

- ✓ Base mathématique solide pour comprendre son fonctionnement.
- ✓ Variabilité de la forme.
- ✓ Alignement temporel incorporé systématiquement.
- ✓ Reconnaissance réalisée cumulée.
- ✓ Décision globale sans obligation d'utiliser des seuils.
- ✓ Séparation franche entre données et algorithmes.

II.3.4.4 Les inconvénients des HMMs :

- ✓ le choix a priori de la typologie des modèles (nombre d'états ,transitions autorisées et règles de transitions).
- ✓ dégradation des performances si l'apprentissage n'est pas suffisant[67].

II.3.4.5 Technique utilisé pour La reconnaissance des caractères à l'aide de HMM :

La reconnaissance optique de caractères peut être appliquée pour reconnaître texte de tout multimédia tel qu'une image, audio, vidéo. La reconnaissance multimédia automatique est basée sur l'ordinatrice application de reconnaissance de la vision et des formes [68]. On peut utiliser traitement d'image, positionnement des caractères, caractère segmentation, réseau de neurones pour résoudre le problème de reconnaissance d'image en texte. En utilisant un HMM, nous pouvons calculer la chaîne des états cachés, basé sur la chaîne d'observation et utilisant la classification algorithme comme l'algorithme de Viterbi ou des algorithmes de compteurs de hmm, on peut trouver le résultat le plus probable.

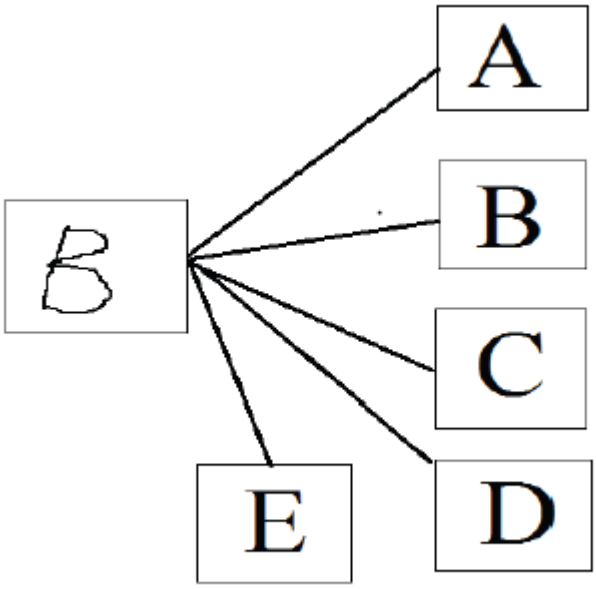


Figure II. 8: *Montrant ce que le logiciel recherche exactement.*

II.3.4.6 Étape à laquelle l'algorithme de contour HMM est appliqué en tant que classificateur pour la reconnaissance :

1. **Training :**

   a- une. Pré-traitement - Traite les données pour qu'elles soient est sous une forme appropriée pour la formation.

   b- Extraction de fonctionnalités - Réduire la quantité des données en extrayant les informations pertinentes Information - résulte habituellement en un vecteur de valeurs scalaires. (Nous avons aussi besoin de normaliser les caractéristiques pour la distance des mesures!).

   c- Estimation du modèle - à partir de l'ensemble fini de vecteurs caractéristiques, besoin d'estimer un modèle (Habituellement statistique) pour chaque classe de données d'entraînement [69].

2. **Testing :**

   a- une. Pré-traitement.

   b-Extraction de caractéristiques.

   c- Classification - Comparer la fonctionnalité des vecteurs à différents modèles et trouver le match le plus proche. On peut utiliser une distance mesuré aussi.

   Dans cet article, nous avons utilisé le processus de classification des motifs. Pour reconnaître les caractères qui est montré ci-dessous dans forme schématique [70] :

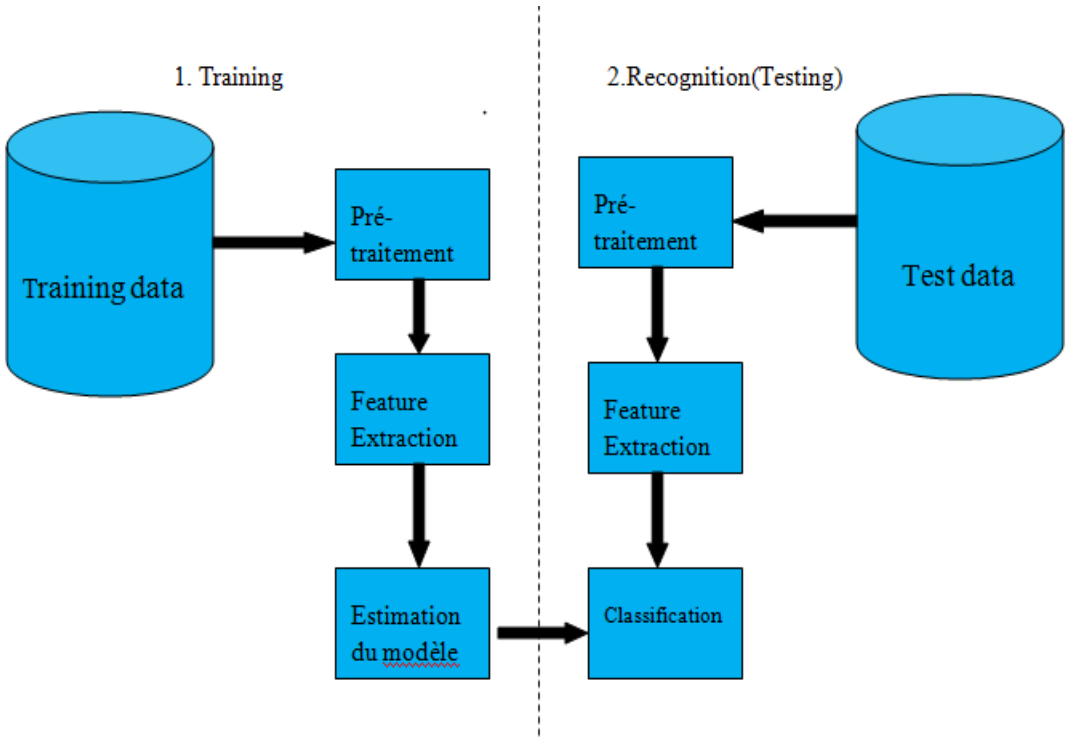


Figure II. 9: *traitement de classification de modèle utilisé.*

Conclusion :

Dans ce chapitre, nous avons abordé les approches de classification plus précisément la supervisé, en donnant les principaux méthodes proposés dans la littérature. Il ne s'agit pas de faire une présentation détaillée de toutes les méthodes mais seulement de préciser les plus classiques que nous utiliserons dans le cadre de notre travail en fonction de leurs propriétés particulières, cette classification est évaluée par une matrice de confusion pour présenter les bonnes décisions.

Nous avons présenté dans ce chapitre un certain nombre de notions et de définitions concernant les outils de reconnaissance que nous allons utiliser pour réaliser notre travail.

CHAPITRE III

IMLPLEMENTATION DES SYSTEMES LPR

III.1 Introduction :

Ce chapitre est le dernier de notre travail, il sera consacré à la mise en œuvre de notre application en indiquant les outils de développement, les langages de programmation utilisés pour son implémentation et les différentes méthodes disponibles pour la reconnaissance des formes que nous avons utilisée dans notre application de détection des plaques d'immatriculation.

Nous présentons ensuite les résultats de nos tests expérimentaux obtenus grâce à notre application.

III.2 Outils de développement :

Parmi les différents outils de développement, nous avons choisi pour réaliser notre travail :

III.2.1 Anaconda : Qu'est-ce que Anaconda?

Anaconda est une distribution Python particulièrement populaire pour l'analyse de données et informatique scientifique Projet Open Source développé par Continuum Analytics, Inc. Disponible pour Windows, Mac OS X et Linux Comprend de nombreux forfaits populaires: NumPy, SciPy, Matplotlib, Pandas, IPython, Cython.

Inclut Spyder, un environnement de développement Python Inclut conda, un gestionnaire de paquets indépendant de la plate-forme. Anaconda est une plateforme logicielle libre comprenant l'environnement Python complet (v2.7 et 3.5) et de très nombreuses librairies (numpy, pylab …). Elle tourne sur tous les OS courants dont Windows [71].

III.2.2 Démarrage d'Anaconda :

A l'issue de l'installation, vous aurez Anaconda Navigator. Ce dernier est une interface graphique permettant de lancer *Jupyter* (pour écrire du python au format notebook), ou encore Spyder (IDE pour éditer du code python).

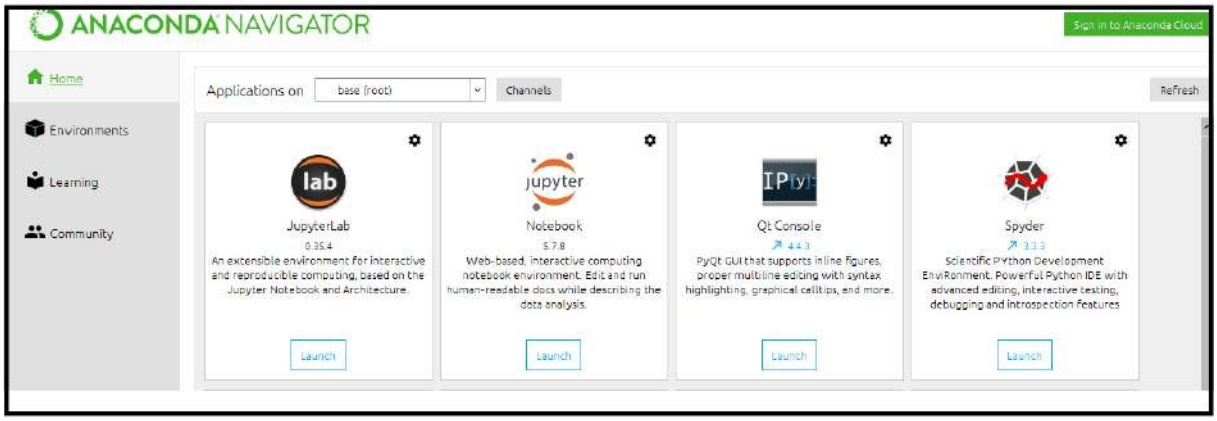


Figure III. 1: *Interface graphique de Anaconda Navigator.*

Depuis cette interface graphique, il est tout à fait possible de gérer les packages installés. Mais pour le moment, on utilisera **conda** : l'invite commande d'Anaconda. Pour cela, on lance le programme "**Anaconda Prompt**" depuis le menu démarrer de Windows comme suit :



Figure III. 2 : *Fenetre de Anaconda prompt.*

III.2.3 Disposer des dernières mises à jour :

Avec la commande conda. Vous pouvez notamment : mettre à jour Python et le gestionnaire de packages Conda avec : conda update conda.



Figure III. 3: *Gestionnaire de package.*

Faire de même pour autres packages utiles pour faire du Machine Learning. Pour s'assurer de la version de conda, tapez : conda –V.

Ensuite assurez-vous que vous avez Python 3.X bien installé : python –V.

L'installation d'un *package* Anaconda spécifique avec : **conda Install** *package*

L'installation de python 3.6 avec la commande : conda Install python=3.6

III.3 Environnement de travail :

III.3.1 Spyder :

Spyder est un environnement de développement pour Python. Il fournit une interface graphique avec une console en bas à droite, des informations au-dessus et un script à gauche (voir figure III.4).

l'explorateur de fichier, puis se placer dans ce répertoire avec Spyder : le répertoire courant de Spyder est affiché en haut à droite. Par défaut, ce répertoire est :

$HOME/.config/Spyder sur Windows.

On peut rentrer le code directement dans la console ou bien utiliser le script de la partie gauche, le sauvegarder et l'exécuter avec la touche **F5**. Par défaut, le script est sauvegardé dans le répertoire courant de Spyder, d'où l'importance d'être dans un répertoire dédié.

La console peut être Python ou IPython, IPython se voulant plus interactif et plus facile d'accès pour le débutant ; on peut créer une nouvelle console du type voulu avec un clic droit sur l'onglet de la console [72].

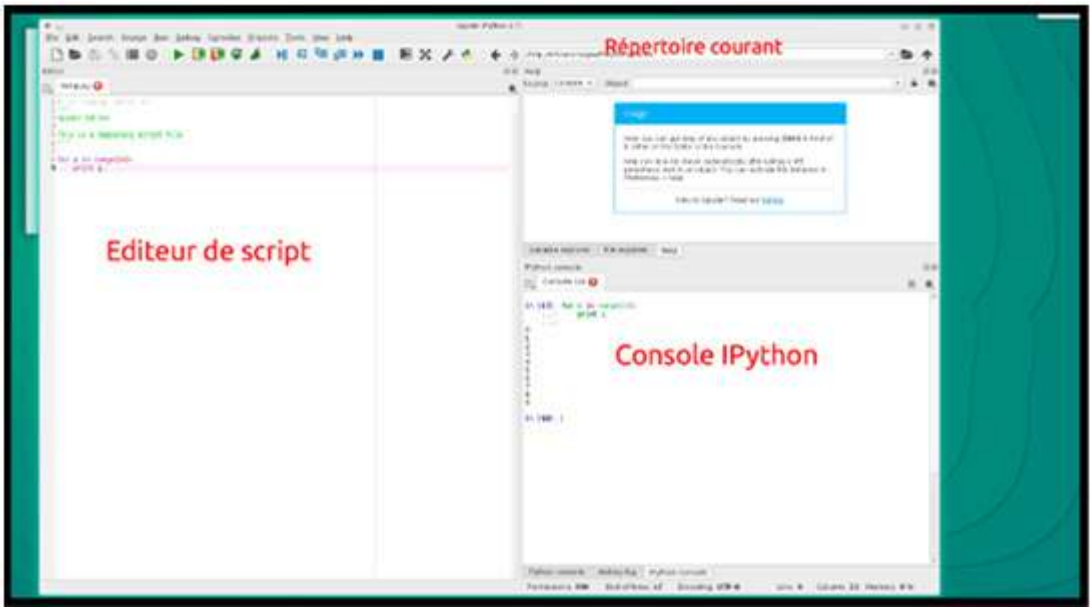


Figure III. 4: *Fenêtre console interactive ipython.*

III.3.1.1 Pourquoi choisir python comme outil de développement :

Ce qui suit incite à choisir python comme développement outil:

1 -Très simple à comprendre et à utiliser.

2 -Présence de modules tiers (paquet Python Indice).

3- bibliothèques de support étendues - Python fournit une grande bibliothèque standard avec peu de fonctions utiles plus adapté au développement de logiciels d'application.

4- langage Python est développé sous un OSI approuvé licence open source, ce qui le rend gratuit pour utiliser et distribuer.

5 -La taille du code est réduite.

6- Python a des structures de données de liste et de dictionnaire intégrées qui peut être utilisé pour construire des données d'exécution rapides structures. En outre, Python offre également la possibilité de typage dynamique des données de haut niveau qui réduit la longueur du code de support nécessaire.

III.3.2 OpenCV :

OpenCV (***Open*** Source ***C***omputer ***V***ision) est une bibliothèque proposant un ensemble de plus de 2500 algorithmes de vision par ordinateur, accessibles au travers d'API pour les langages C, C++, et Python. Elle est distribuée sous une licence BSD (libre) pour les plate-formes Windows, GNU/Linux, Android et MacOS [73].

OpenCV est aujourd'hui développée, maintenue, documentée et utilisée par une communauté de plus de 40 000 membres actifs. C'est **la** bibliothèque de référence pour la vision par ordinateur, aussi bien dans le monde de la recherche que celui de l'industrie, pour java deux adaptation du projet JavaCV ont été faite afin de permettre son utilisation, la première est un artifact du projet OpenCV dans la repository de Maven, la deuxième est un artifact bytedeco qui se trouve aussi dans la repository Maven : ce sont deux interface DLL Différente qui se sert du même noyau qui est OpenCV sous c++.

Afin de mieux vous présenter son étendue et ce qu'elle permet de faire, jetons un œil aux principaux modules accessibles :

- **Core** : les fonctionnalités de base.

Cette bibliothèque permet de manipuler les structures de base, réaliser des opérations sur des matrices, dessiner sur des images, sauvegarder et charger des données dans des fichiers XML…

- **imgproc** : traitement d'image.

Nous entrons dans le cœur du sujet. Les fonctions et structures de ce module ont trait aux transformations d'images, au filtrage, à la détection de contours, de points d'intérêt…

- **features2d** : descripteurs.

Ce module concerne principalement l'extraction de descripteurs selon deux approches courantes (SURF et StarDetector), que nous aborderons lorsque nous nous intéresserons à la caractérisation d'images.

- **objdetect** : détection d'objets.

Cette bibliothèque permet de faire de la reconnaissance d'objets dans une image au moyen de l'algorithme Adaboost (Viola & Jones, 2001). Nous y reviendrons lorsque nous parlerons d'apprentissage et de reconnaissance de formes.

- **video** : traitement de flux vidéo.

Ces fonctions servent à segmenter et suivre les objets en mouvement dans une vidéo.

- **Highgui** : entrées-sorties et interface utilisateur.

OpenCV intègre sa propre bibliothèque haut-niveau pour ouvrir, enregistrer et afficher des images et des flux vidéo. Celle-ci contient aussi un certain nombre de fonctions permettant de réaliser des interfaces graphiques très simples, mais largement suffisantes pour tester nos programmes.

- **calib3d** : calibration, estimation de pose et stéréovision.

Ce module contient des fonctions permettant de reconstruire une scène en 3D à partir d'images acquises avec plusieurs caméras simultanément.

III.3.3 L'interface graphique (GUI) :

GUI est une application de bureau qui vous aide à interagir avec les ordinateurs. Ils sont utilisés pour effectuer différentes tâches sur les ordinateurs de bureau, ordinateurs portables, autres appareils électroniques, etc., ici, nous parlons principalement des ordinateurs portables et des ordinateurs de bureau.

Les applications graphiques telles que Text-Editors sont utilisées pour créer, lire, mettre à jour et supprimer différents types de fichiers.

Les applications graphiques telles que le sudoku, les échecs, la solitaire, etc., sont des jeux auxquels vous pouvez jouer.

Les applications graphiques telles que Chrome, Firefox, Microsoft Edge, etc., sont utilisées pour surfer sur Internet.[74]

III.3.4 tkinter library :

Tkinter est le module Python specifiques aux interfaces graphiques (GUI) fourni par défaut. Il dérive de Tk (ToolKit) une extension graphique du langage de script Tcl (Tool Command Language). Et il est utilisé pour créer des applications simples à interface graphique. C'est le module le plus couramment utilisé pour les applications à interface graphique en Python.

III.3.5 Remarque:

Je vais utiliser la version 3.6 de Python. Alors, veuillez mettre à jour Python Supposerons bien évidemment que le module Tkinter a déjà été installé sur votre système. Pour pouvoir en utiliser les fonctionnalités dans un script Python, il faut que l'une des premières lignes de ce script contienne l'instruction d'importation [74] :

<< from Tkinter import >>

Dans notre projet nous avons besoin d'installer les modules suivant :

- import xlrd : Pour la lecture des fiché exel
- importe xlwt : Pour l'écriture dans le fichier exel
- importe datatimes :Pour affichiez le temp
- ipykernel pipi mage ; Pour lire les image.

III.3.6 Conclusion

Nous avons présenté un certain nombre de notions et de définitions concernant les outils de développements que nous avons utilisé pour réaliser notre travail. La partie suivante de ce chapitre détaillera les étapes de l'implémentation de notre application.

III.4 IMPLEMENTATION

III.4.1 Introduction :

Après avoir vu les différents outils de développement de notre système, nous passons à l'implémentation de l'application définies et détaillée dans ce chapitre en expliquant les écrans de notre système.

III.4.2 Architecture générale :

Le système ce compose de deux étage essentiel, le premier concerne l'algorithmique du traitement d'image et les étapes de segmentation et le conditionnement de l'image pour pouvoir extraire du caractère de l'image (OCR) et le second concerne la gestion temps réel de l'application avec la base de donnée :

Pour qu'un système soit évolutif, extensible et maintenable, une bonne architecture doit être choisie. La figure ci-dessous montre l'architecture de notre système.

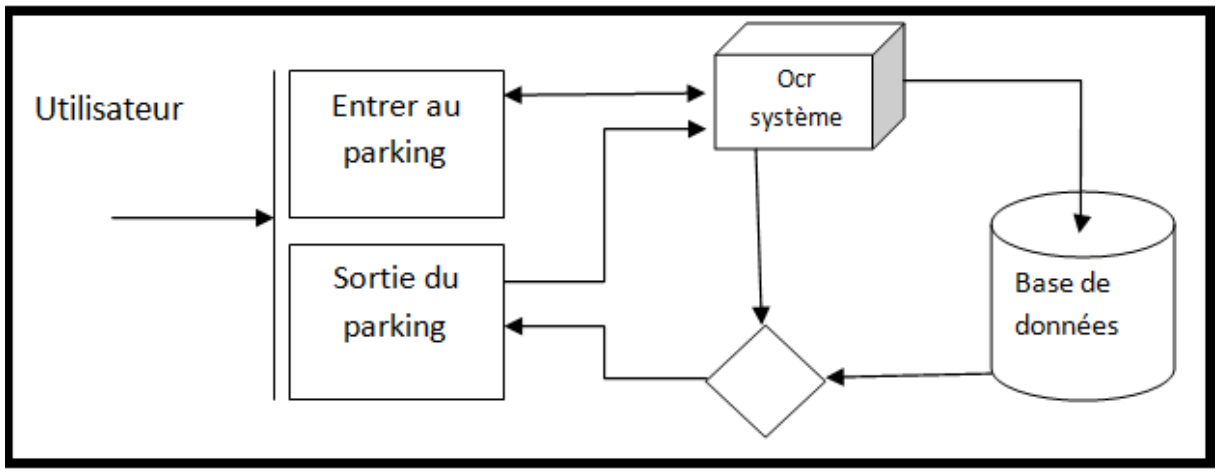


Figure III. 5: *Architecture générale du système.*

III.4.3 Fonctionnement :

Pour mieux comprendre le fonctionnement de l'application, les figures qui suivent décrivent la chronologie des principales fonctionnalités de l'application.

Le système au premier lieux s'affiche comme une simple interface graphique avec deux bouton entrer et sortie , et deux champ d'affichage , le bouton entrer récupère une capture temps réel de caméra et le fait passer par le système (OCR) pour extraire le matricule en même temps le matricule et enregistré dans une base de donnée dans deux champ le premier pour le matricule en question et en parallèle le jour la date et l'heur d'entrer .

Le premier champ d'affichage affiche le matricule pour permettre a l'utilisateur de vérifier si la machine fait bien son travail.

Le deuxième bouton dans le cas ou la voiture sort du parking l'utilisateur utilise le bouton sortie pour que le système fait le même travail précédent mais cette fois il compare avec la base de donnée l'ID correspondant a cette voiture en parallèle il récupère le temps du système pour calculer le prix et l'afficher, voire (figure III.6) et (figure III.7) :

III.4.3.1 Les interfaces de l'application :



Figure III. 6: U*n capture montrant la phase d'entrer au parking.*



Figure III. 7:*Une capture de l'interface montrant la phase de sortie du parking.*

- **Création dune basse dé donne avec python :**

Nous avent importé le module xlrd :pour lire les fichie Excel de puis python Et le module xlwt pour la l'écrite dans de fichié Excel ;

Voici la partie du code responsable :

```
33 THIS_FOLDER = os.path.dirname(os.path.abspath(__file__))
34 my_file = os.path.join(THIS_FOLDER, 'database1.txt')
35 file = open("database1.txt", "w")
```

- **Le calcule de prix**

Aprés la récupère de temps le système pour calculer le prix nous avant ajouté cette fontion:

```
H_P= heur_now-heur_lu
if H_P<0:
    H_P=0

M_P= (minut_now-minut_lu)/60

heur_total=H_P+M_P
prix =heur_total*100
prix=str(prix)+' DA'
self.varB.set(prix)
```

III.4.4 Démarche suivie :

Plusieurs paramètres ont été pris en compte afin d'étudier leur influence sur la reconnaissance et la lecture des plaques.

III.4.4.1 Caméra :

Un système de caméras également été étudié en vue d'applications utilisant plusieurs points de vues à surveiller**.**

Pour cela on a utilisé une webcam pour capturer et diffuser des images en temps réel sur tous types.

Les photos prises sont alors directement transférées sur le disque dur du PC via une liaison USB est enregistrées au format png avec une résolution de 640x480 et en couleur.

III.4.4.2 Couleur ou Noir et Blanc :

La couleur n'a aucune influence sur les traitements d'images effectués dans la localisation du mobile ou dans la lecture des caractères.

III.4.4.3 Position de la caméra :

L'axe optique de la caméra doit-il être parallèle à l'axe de la chaussée. C'est un choix qui affectera les performances de l'OCR.

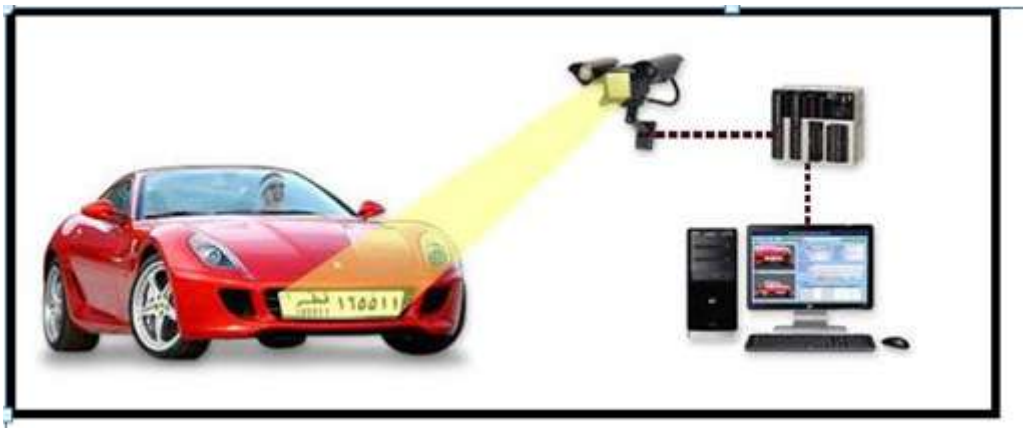


Figure III. 8: *Reconnaissance des plaque de matriculassions.*

- **Les étapes Algorithmique : Recherche d'une plaque d'immatriculation à l'aide de K-NN :**

1: commence

2: Entrée: Image originale

3: sortie: caractères

4: Méthode: K-voisins les plus proches

5: LP: plaque d'immatriculation

6: Convertir une image RVB en niveaux de gris

7: filtrer la transformation morphologique

8: Transforme une image en niveaux de gris en image binaire

9: Filtre gaussien pour brouiller l'image

10: Trouver tous les contours dans l'image

11: Rechercher et reconnaître tous les caractères possibles dans l'image

12: Rogner une partie de l'image avec le plus haut candidat LP

13: Recadrer le LP à partir de l'image originale

14: appliquer à nouveau les étapes 6 à 11 sur l'image de recadrage

15: Imprimer les caractères dans le LP

16: fin

III.4.5 Étapes de la reconnaissance des plaques d'immatriculation avec python :

La reconnaissance de plaque d'immatriculation ou LPR, en abrégé, comporte trois étapes principales. Les étapes sont les suivantes :

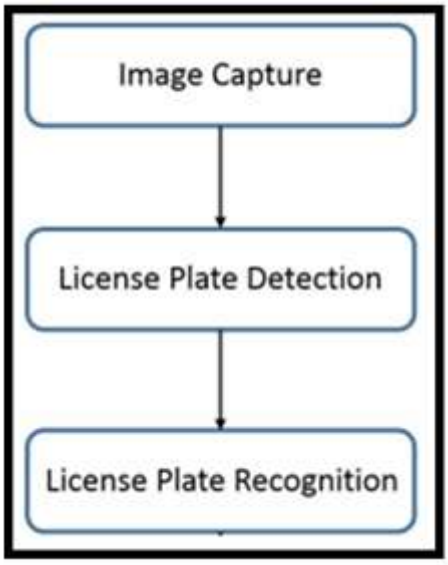


Figure III. 9: *Étapes typiques d'un système RAPI.*

III.4.5.1 License Plate Détection :

La première étape consiste à détecter la plaque d'immatriculation de la voiture. Nous utiliserons l'option de contour dans OpenCV pour détecter les objets rectangulaires afin de trouver la plaque d'immatriculation. La précision peut être améliorée si nous connaissons la taille exacte, la couleur et l'emplacement approximatif de la plaque d'immatriculation. L'algorithme de détection est formé en fonction de la position de la caméra et du type de plaque d'immatriculation utilisé. Cela devient plus compliqué si l'image ne contient même pas de voiture. Dans ce cas, nous effectuerons une étape supplémentaire pour détecter la voiture, puis la plaque d'immatriculation.

III.4.5.2 Character Segmentation:

Une fois que nous avons détecté la plaque d'immatriculation, nous devons la rogner et la sauvegarder en tant que nouvelle image. Encore une fois, cela peut être fait facilement avec OpenCV.

III.4.5.3 Reconnaissance des caractères:

Maintenant, la nouvelle image obtenue à l'étape précédente est sûre d'avoir certains caractères (Nombres / Alphabets) écrits dessus. Ainsi, nous pouvons effectuer une reconnaissance optique de caractères (OCR) pour détecter le numéro. Nous avons déjà expliqué La reconnaissance optique de caractères (OCR) avec k-NN.

III.4.6 Architecture et conception d'Ocr :

Dans cette architecture trois problème sont traité dès le début du traitement nous avons classé en trois grande titre. au tout début la détection des plaque dans la scène en deuxième partie la détection des caractère dans les plaques, au finale une classification a base du knn est effectué nous allons détailler chaque processus indépendamment et nous commençons avec la détection des plaques.

III.4.7 La détection des plaques :

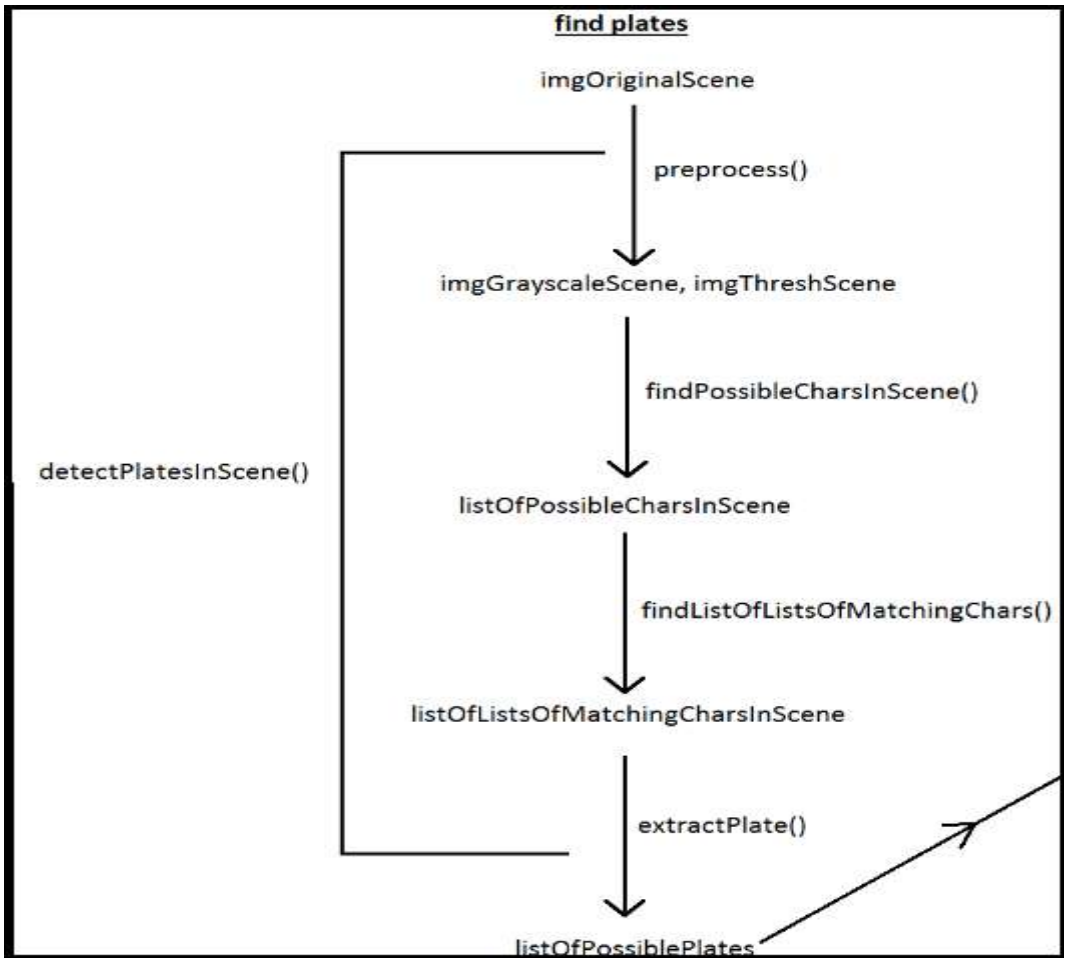


Figure III. 10: *L'architecture de notre système OCR.*

1) Phase de traitement d'image

a) Prétraitement :

Prenons un exemple d'image d'une voiture et commençons par détecter la plaque d'immatriculation de cette voiture. Nous utiliserons ensuite la même image pour la segmentation et la reconnaissance de caractères, L'image de test que nous utilisons pour ce tutoriel est présentée ci-dessous : (Figure III.11).

- **Capture d'image :** À partir du premier point de la capture de l'image, OpenCV la bibliothèque a des algorithmes hautement optimisés pour toutes les images opérations de traitement. OpenCV fournit une interface pour différents modèles de caméras. L'extrait de code suivant explique comment interfacer une caméra Web intégrée et capturer un cadre :

```
from opencv import highgui as hg
capture = hg.cvCreateCameraCapture(0)
hg.cvNamedWindow("Snapshot")
frame = hg.cvQueryFrame(capture)
hg.cvShowImage("Snapshot", frame)
```



Figure III. 11: *Image original RGB.*

Conversion de l'image en niveau de gris

Cette étape consiste à transformer l'image originale du plant RGB € $N^{n.m.3}$ a une image grayscale € $N^{n.m}$ la raison de cette phase c'est pour optimiser le traitement de 33.33% et d'une autre part conditionner la possibilité a la binairisation, voici la partie du code :

```
img = cv2.resize(img, (620,480) )
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) #convert to grey scale
```

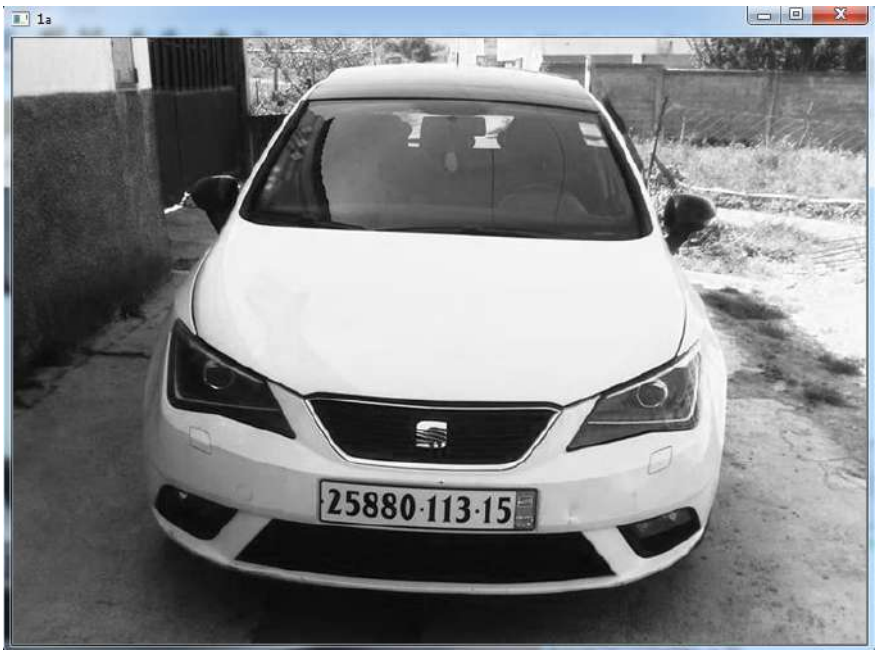


Figure III. 12: *Image a niveau de gris.*

- **Transformation de l'image en binaire**

Après le passage a l'image a niveau de gris il faut binairiser l'image, la méthode utiliser est le Seuil adaptatif avec moyenne pondérée gaussienne qui ce classe parmi les algorithmes de seuillage automatique comme l'algorithme d'Otsu :

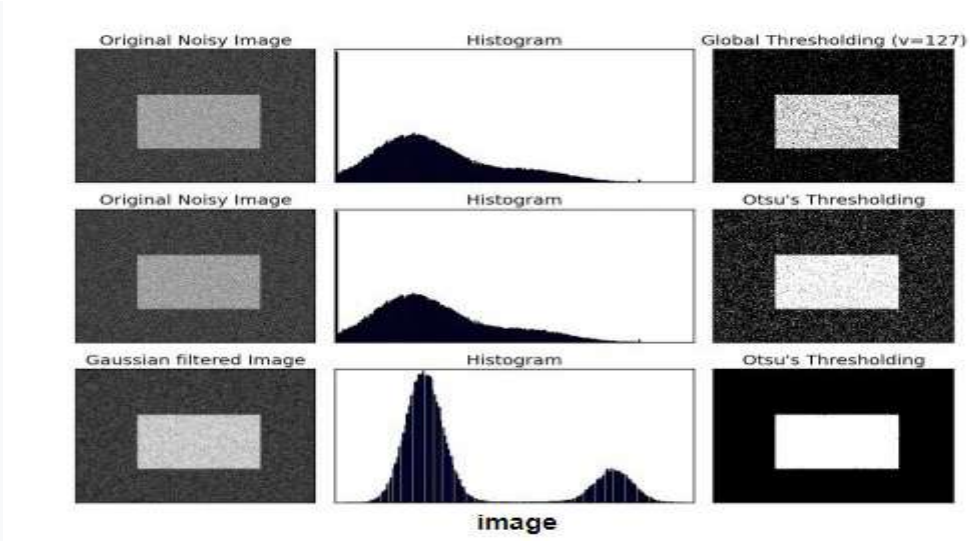


Figure III. 13: *Image explicatif sur le seuillage automatique a base d'histogramme.*

- **Suppression ou diminution du bruit**
  - **Filtre gaussien :** C'est également un filtre passe-bas. Une gaussienne à deux dimensions est donnée par l'expression suivante :

$$g(x,y) = \frac{1}{2\Pi\sigma^2} exp(-\frac{x^2+y^2}{2\sigma^2})$$

Le filtre gaussien donne un meilleur lissage et une meilleure réduction du bruit que le filtre moyenne. En général un filtre gaussien avec σ< 1 est utilisé pour réduire le bruit, et si σ>1 c'est dans le but de fabriquer une image qu'on va utiliser pour faire un masque flou personnalisé. Il faut noter que plus σ est grand, plus le flou appliqué à l'image sera marqué.

Voici la partie du code responsable à la binairisation :

```
imgBlurred = cv2.GaussianBlur(imgMaxContrastGrayscale, GAUSSIAN_SMOOTH_FILTER_SIZE, 0)

imgThresh = cv2.adaptiveThreshold(imgBlurred, 255.0, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY_INV,

return imgGrayscale, imgThresh
```

Voici le résultat obtenue (Figure III.14) :



Figure III. 14: *Image binaire par la méthode Seuil adaptatif avec moyenne pondérée gaussienne.*

b) Trouver toutes les plaques possibles :

B-1 Phase pour la détection des contours :

Les contours peuvent être expliqués simplement comme une courbe joignant tous les points continus (le long de la limite), ayant la même couleur ou intensité. Les contours sont un outil utile pour l'analyse de forme, la détection et la reconnaissance d'objets.

- Pour une meilleure précision, utilisons des images binaires. Donc, avant de trouver des contours, appliquez une détection de seuil ou de bord malicieux.

- La fonction findContours modifie l'image source. Donc, si vous voulez une image source même après avoir trouvé des contours, stocker la déjà dans d'autres variables.

- Dans OpenCV, rechercher des contours revient à rechercher un objet blanc sur un fond noir. Rappelons-nous, l'objet à trouver doit être blanc et l'arrière-plan doit être noir comme dans notre cas les nombres de matricule sont en blanc voir figure III.15.

Voici la partie du code qui fait ce travail :

```
imgThreshCopy = imgThresh.copy()

# imgContours, contours, npaHierarchy = cv2.findContours(imgThreshCopy, cv2.RETR_TREE, cv2.CHAIN_APP
contours, npaHierarchy = cv2.findContours(imgThreshCopy, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
```

Voici le résultat obtenu (figure III.15).



Figure III. 15: *l'image binaire introduite sur contour founding.*

Après il faut extraire les caractères possibles sur la scène pour créer un vecteur avec la fonction qui gère ce processus est :

```
self.boundingRect = cv2.boundingRect(self.contour)
```

Voici le résultat obtenu :

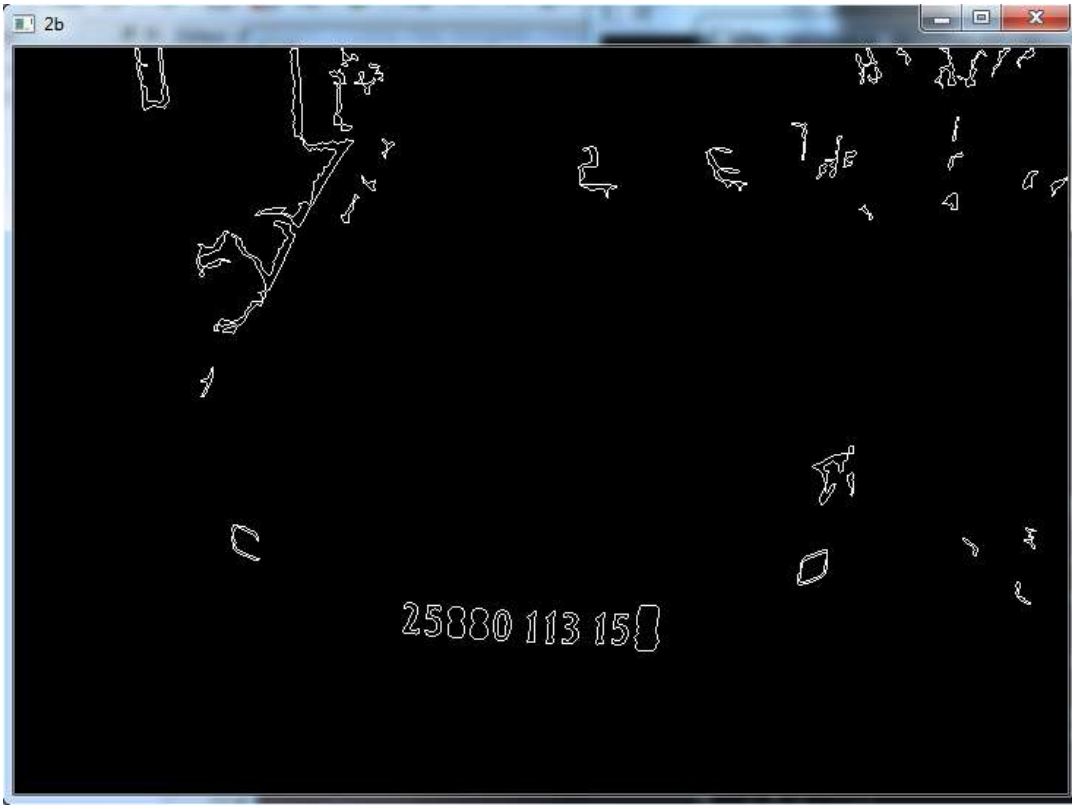


Figure III. 16: *Image filtré par rapport au caractère existant.*

Nous pouvons donc masquer l'ensemble de la photo, à l'exception de l'endroit où se trouve la plaque d'immatriculation. Le code pour faire la même chose est indiqué ci-dessous :

```
def detectCharsInPlates(listOfPossiblePlates):
    intPlateCounter = 0
    imgContours = None
    contours = []

    if len(listOfPossiblePlates) == 0:          # if list of possible plates is empty
        return listOfPossiblePlates             # return
```

La nouvelle image masquée apparaîtra comme ci-dessous :



Figure III. 17 : *les caractères détectés de la plaque.*

Délimitation :

Tout le traitement précèdent a été réalisé en préparation du terrain pour cette étape qui va servir à OpenCV (CvFindContour) à scanner l'image et à détecter les contours qui sont bien claire grâce à la dilatation et bien détectable grâce à Canny. Nous ignorons les petits rectangles détectés, car une plaque à d'importantes dimensions par rapport à la proportion de l'image. A la fin nous ne gardons que les rectangles de tailles importantes afin de les fournir au lecteur OCR qui va se servir pour la lecture

c) La détection des caractères :

c-1) Phase de lecture OCR

Cette phase rassemble des techniques qui réalisent une reconnaissance de la forme des caractères :

- **Prétraitement :** La prochaine étape de la reconnaissance des plaques d'immatriculation consiste à segmenter la plaque d'immatriculation en la coupant et en l'enregistrant comme une nouvelle image. Et faire les mêmes étapes précédentes, voir la figure si dessous :



Figure III. 18: *image originale coupé.*

1) **Étape 1** : Convertissez l'image originale en image en niveaux de gris :



Figure III. 19: *image a niveau de gris.*

2) **Étape 2:** convertissez l'image en niveaux de gris obtenue à l'étape 1 en image binaire et appliquez-la transformation morphologique à l'aide de la bibliothèque OpenCV pour détecter facilement les LP.



Figure III. 20:image binairisée.

3) **Étape 3 :** Appliquez un filtre flou gaussien pour lisser l'image Figure III.22.



Figure III. 21: *Lissage de l'imagepar le filtre Gaussien.*

4) **Étape 4 :** Réperté tous les contours de l'image pour supprimer les limites de l'image



Figure III. 22:*Liste de tous les contours dans l'image.*

5) **Étape 5:** Utilisation de l'algorithme KNN pour répertorier tous les caractères correspondants dans l'image.



Figure III. 23: *Toute la plaque possible qui peut être trouvé.*

6) **Etape 6 :** La dernière étape de cette reconnaissance de plaque d'immatriculation consiste à lire les informations de plaque d'immatriculation à partir de l'image segmentée :

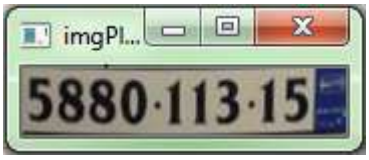


Figure III. 24: *Extracted plate.*

- ***Lecture ocr :***

Finalement une lecture est effectuée est le résultat est affiché à droite.

Voir la figure 26

d) Les résultants de simulation par python:

La plupart du temps, la qualité et l'orientation de l'image sont correctes, le programme a pu identifier la plaque d'immatriculation et en lire le numéro. Les clichés ci-dessous montrent peu de résultats positifs.

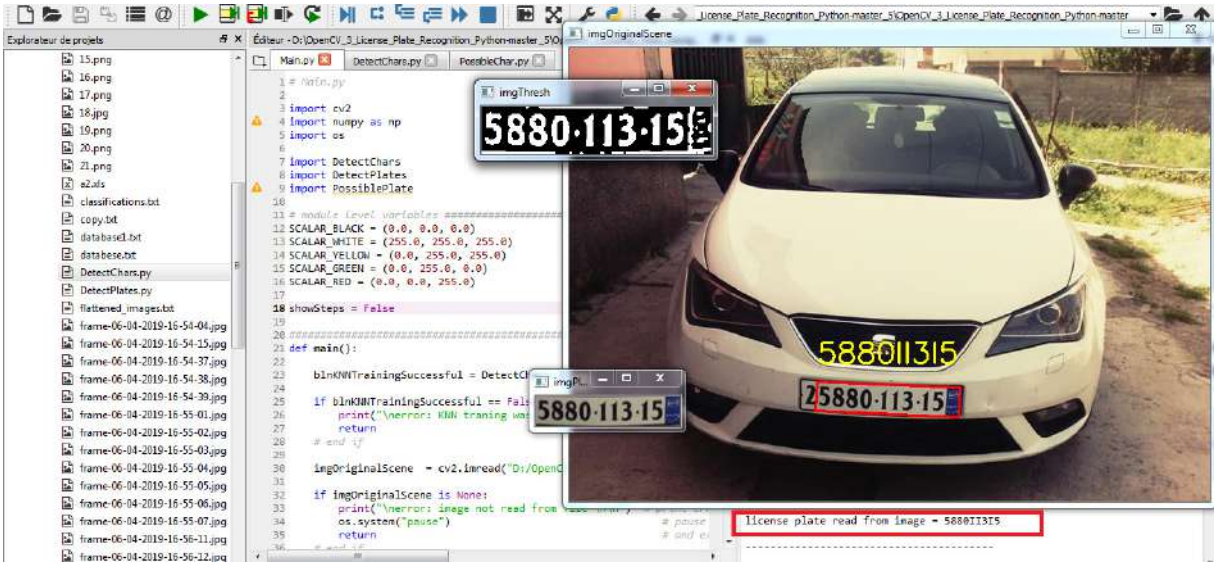


Figure III.26: *Identification de la plaque d'immatriculation.*

Comme vous pouvez le voir, l'image originale portait le numéro «2588011315» et notre programme a détecté la même valeur à l'écran.

❖ **Échecs dans la reconnaissance de plaques d'immatriculation :**

Les résultats de cette méthode ne seront pas précis. La précision dépend de la clarté de l'image, de l'orientation, de l'exposition à la lumière, etc. Pour obtenir de meilleurs résultats, if fallait d'implémenter des algorithmes d'apprentissage automatique.

Pour avoir une idée, regardons un autre exemple où la voiture ne fait pas face à la caméra directement :

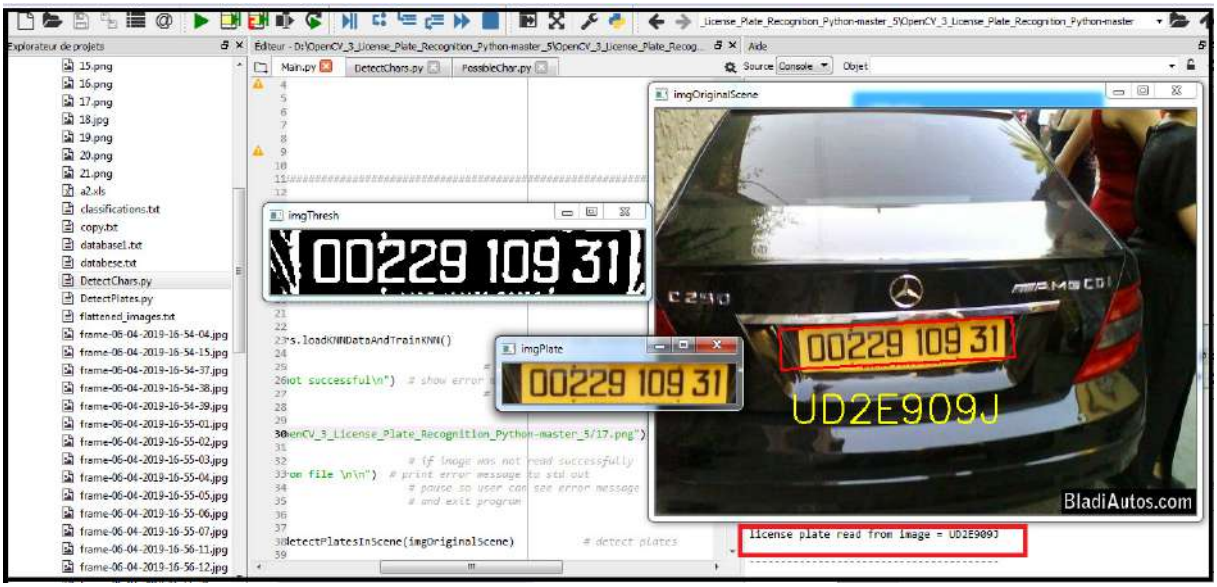


Figure III. 25: *Exemple d'un échec de reconnaissance.*

Comme vous pouvez le constater, notre programme a pu détecter correctement la plaque d'immatriculation et la rogner. Mais la bibliothèque n'a pas réussi à reconnaître les caractères correctement. Au lieu du «0022910931», l'OCR l'a reconnu comme <<ud2e909j>> Des problèmes comme celui-ci peuvent être corrigés en utilisant des images mieux orientées.

Dans le pire des cas, le contour ne parvient pas à détecter correctement la plaque d'immatriculation. L'image ci-dessous contient trop d'informations d'arrière-plan et d'éclairage insuffisant pour que le programme n'ait même pas réussi à identifier la plaque d'immatriculation à partir du numéro. Dans ce cas, nous devons encore relayer sur l'apprentissage machine ou améliorer la qualité de l'image.

e)Autres exemples réussite :



Figure III. 26: *Reconnaissance de plaque réussite.*

La plupart du temps, la qualité et l'orientation de l'image sont correctes, le programme a pu identifier la plaque d'immatriculation et en lire le numéro. Les clichés ci-dessous montrent peu de résultats positifs. Encore une fois.

En prendre un autre exemple une image qui contient un matricule alphanumérique :

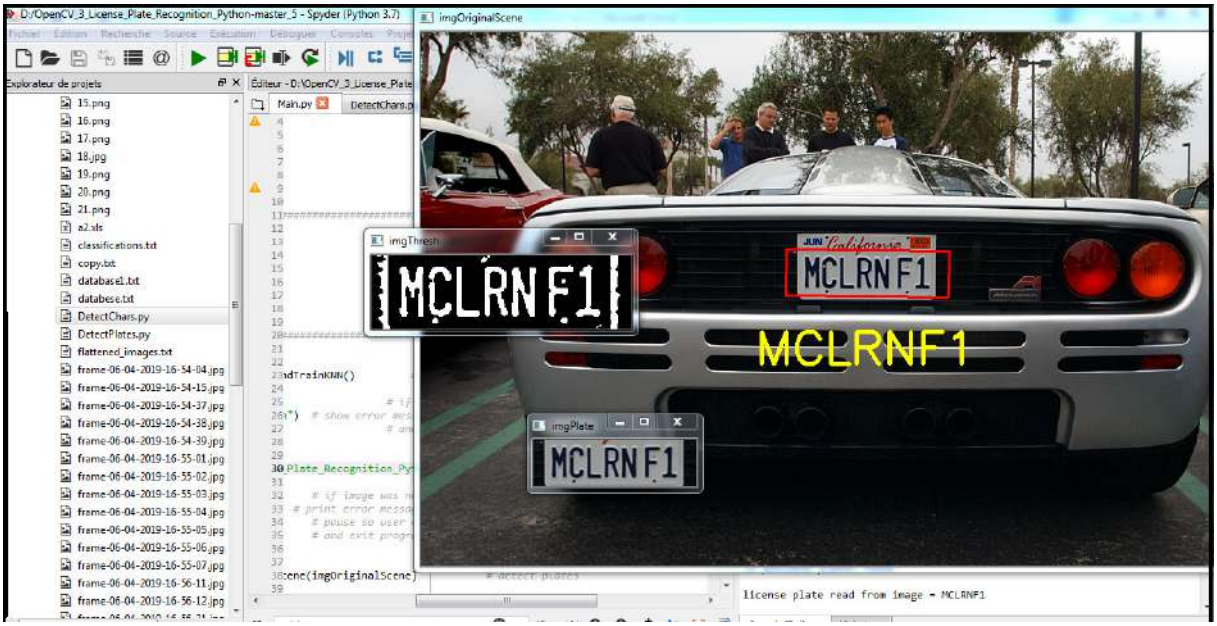


Figure III. 27: *Matricule alphanumérique.*

III.5 Résultats expérimentaux :

Le message de cette recherche est de montrer que les technologies open source sont suffisamment matures pour la recherche scientifique et les domaines informatiques. Python et OpenCV sont de bons points de départ pour les chercheurs et les étudiants en vision par ordinateur.

Notre expérience est basée sur un processeur Intel Core i5 à 2,80 GHz avec 8 Go de RAM

et mis en œuvre en utilisant C avec la bibliothèque OpenCV. De 30 images qui ont pris en condition différente utilisée pour tester la méthode proposée 22 images environ 91,5% des images détectés avec précision. Il y a quelques raisons qui conduisent à une licence non reconnue.

La première raison est si la plaque d'immatriculation est sale et pas claire. Une autre raison s'il n'y a pas assez d'éclairage sur la plaque d'immatriculation durant la nuit. De plus, il est difficile de détecter le matricule.

Et la qualité d'image de camera : le système affiche une précision de 100% pour une image de haute qualité, 82,6% pour une qualité moyenne et 44,5% pour une image de faible qualité,

Conclusion :

Dans Ce chapitre présente une technique proposée pour détecter et reconnaître la plaque d'immatriculations basée sur l'algorithme K- NN. L'image de la plaque d'immatriculation d'abord prétraitée en plusieurs étapes pour spécifier l'emplacement de la plaque dans l'image puis en coupant partie de l'image. Ensuite, selon les caractéristiques de la plaque d'immatriculation algérien l'algorithme proposé sépare les caractères et les nombres dans la plaque, et reconnaître chaque personnage. Le système proposé a été testé Les résultat a montré que la technique conçue est robuste et qu'elle exécute 91,5%.

CONCLUSION GENERALE

Conclusion générale :

L'objectif de notre travail est de développer un système de détection et de reconnaissance de plaques minéralogiques ayant la capacité d'extraire les numéros d'immatriculation. Après avoir présenté un état de l'art sur les systèmes existants, nous avons présenté notre système LPR (Licence plate recognition) où nous avons réussi à détecter avec une grande précision les chiffres formant le matricule de véhicule. Le travail suivant représente le fruit d'une recherche dans le domaine de la reconnaissance des plaques d'immatriculation, dans lequel nous nous sommes intéressés à extraire les caractéristiques des plaques en utilisant la méthode de classification K-NN Premièrement nous avons exposé les différentes étapes de prétraitement contenant la phase de détection de la de la plaque, ainsi qu'une étape de segmentation, après l'isolation des caractères de la plaque Ensuite nous avons introduit la notion de k plus proche voisin pour extraire les caractéristiques de l'image caractère.

Pour tester et évaluer notre système de reconnaissance, nous avons utilisé python comme outil de développement Les résultats de reconnaissance obtenus montrent l'impact de la méthode classification k-NN sur la reconnaissance et prouvent si efficacité.

REFERENCES

[1] B.V.Dhandra, R. M. (2007.). *Handwritten Character Recognition using K-NN Classification Algorithm.* Handwritten Kannada Numeral Recognition Based on Structural .

[2] LeCun, Y. *http://www.research.att.com/yann/exdb/mnist/index.html.* NewYork: Courant Institute, NYU

Corinna Cortes.

[3] Macker, S. a. (Jan.1999.). *Routing Protocol Performance Issues and Evaluation Considerations.* Naval Research Laboratory.

[4] Professor, G. R. *K Nearest Neighbour Classification over Encrypted Relational Data.* Sangamner, Maharashtra,India.: 1,2Department of Information Technology Engineering.

[5] Singh, D. K. *Text Recognition from Images.* India.: Department of Computer Technology,.

[6] Singh, D. K. *Text Recognition from Images.* India.: Department of Computer Technology.

[7] Un modèle de Markov caché est un processus doublement stochastique. (February - 2014). *Implementation of Character Recognition using.* International Journal of Engineering Research & Technology.

Yusoff, M. I. (November 18 – 20, 2014,). *Hidden Markov Models.* Putrajaya, Malaysia .

[8] A. Zidouri, M. Sarfraz :" On optical character recognition of Arabic text ". The 6th Saudi Engineering Conference, Volume 4, KFUPM, Dhahran, December 2002.

[9] J. Y. Toumit, S. G. Salicetti, H. Emptoz :" Caractérisation d'objets mathématiques et redondance graphique pour la lecture automatique de documents mathématiques ". Vision Interface '99, Trois-Rivières, Canada, 19-21 May.

A. Belaïd :" Reconnaissance automatique de l'écriture et du document ". LORIA-CNRS, Campus scientifique B.P. 239, 54506 Vandoeuvre-Lès-nancy, France.

[10] G. Stamon, N. Vincent :" Bio-Imagery – Image Analysis Document Analysis – Pattern Recognition ". Rapport d'activité SIP, Laboratoire CRIP5, Université Paris Descartes, France, Avril 2008.

[11] M. Côté :" Utilisation d'un modèle d'accès lexical et de concepts perceptifs pour la reconnaissance d'images de mots cursifs ". Thèse de Docteur de l'école Nationale Supérieur de Télecommunications, France, Juin 1997.

[12] J. André:" Courrier: Histoire d'un caractère de la machine à écrire aux fontes numériques ". Version fac-similée, France, Janvier 2010.

[13] J.M.DOUDOUX, "Maven", Développons en Java, Copyright(c) 1999-2016.

[14] M.G Gauthier, "GitHub, u'est-ce que c'est ? ", OpenClassrooms, 12 décembre 2016-Europe.

[15] M.DALMAU, " Ecriture d'application Java avec swing ", IUT de Bayonne, 2013-Lyon.

[16] A.Calmettes, "Introduction à la vision par ordinateur", OpenClassrooms, 5 décembre

[17] M. Kumar Vijay, Cross-Disciplinary Applications of Artificial Intelligence and Pattern Recognition: Advancing Technologies: Advancing Technologies. IGI Global, 2011.

[18] A.-H. Mohammad Ali, Advanced Techniques in Multimedia Watermarking: Image, Video and Audio Applications: Image, Video and Audio Applications. IGI Global, 2010.

[19] H. Zhu, H. Shu, T. Xia, L. Luo, et J. Louis Coatrieux, « Translation and scale invariants of Tchebichef moments », Pattern Recognit., vol. 40, no 9, p. 2530-2542, sept. 2007.

[20] S. N. Yaakob, P. Saad, et M. F. Jamlos, « On Analysis of Invariant Characteristic for Moment Invariant Techniques », 2006.

[21] R. Mukundan, « Some computational aspects of discrete orthonormal moments », Image Process. IEEE Trans. On, vol. 13, no 8, p. 1055-1059, 2004.

[22] P. Comon, « Classification supervisée par réseaux multicouches », Trait. Signal, vol. 8, n o 6, p. 387-407, 1991.

[23] J. de Villiers et E. Barnard, « Backpropagation neural nets with one and two hidden layers », IEEE Trans. Neural Netw., vol. 4, no 1, p. 136-141, janv. 1993.

[24] H. Bhavsar et A. Ganatra, « A comparative study of training algorithms for supervised machine learning », Int. J. Soft Comput. Eng. IJSCE, vol. 2, no 4, p. 2231-2307, 2012.

[25] B. El Kessab, C. Daoui, et B. Bouikhalene, « Moroccan-style PlateS Recognition Based on Support Vectors Machines », 2015.

[26] S. Kim, D. Kim, Y. Ryu, et G. Kim, « A robust license-plate extraction method under complex image conditions », in 16th International Conference on Pattern Recognition, 2002. Proceedings, 2002, vol. 3, p. 216-219 vol.3.

[27] Y. Zheng, H. Li, et D. Doermann, « Machine printed text and handwriting identification in noisy document images », IEEE Trans. Pattern Anal. Mach. Intell., vol. 26, no 3, p. 337-353, mars 2004.

[28] Z. Bin, L. Yong, et X. Shao-Wei, « Support vector machine and its application in handwritten numeral recognition », in 15th International Conference on Pattern Recognition, 2000. Proceedings, 2000, vol. 2, p. 720-723 vol.2..

[29] R. G. Wijnhoven et P. H. De With, « Identity verification using computer vision for automatic garage door opening », Consum. Electron. IEEE Trans. On, vol. 57, no 2, p. 906-914, 2011.

[30] T.-D. Chen, « License-plate recognition using DWT and neural network », Int. J. Wavelets Multiresolution Inf. Process., vol. 4, no 4, p. 601-615, 2006.

[31] R. Mukundan, S. H. Ong, et P. A. Lee, « Image analysis by Tchebichef moments », Image Process. IEEE Trans. On, vol. 10, no 9, p. 1357-1364, 2001.

[32] R. Y. Wong et E. L. Hall, « Scene matching with invariant moments », Comput. Graph. Image Process., vol. 8, no 1, p. 16-24, août 1978. [29] M. Kumar Vijay, Cross-Disciplinary Applications of Artificial Intelligence and Pattern Recognition: Advancing Technologies: Advancing Technologies. IGI Global, 2011.

[33] A.-H. Mohammad Ali, Advanced Techniques in Multimedia Watermarking: Image, Video and Audio Applications: Image, Video and Audio Applications. IGI Global, 2010.

[34] H. Zhu, H. Shu, T. Xia, L. Luo, et J. Louis Coatrieux, « Translation and scale invariants of Tchebichef moments », Pattern Recognit., vol. 40, no 9, p. 2530-2542, sept. 2007.

[35] S. N. Yaakob, P. Saad, et M. F. Jamlos, « On Analysis of Invariant Characteristic for Moment Invariant Techniques », 2006.

[36] F.Biadsy, SEl-Sana , N.Habash, "Online Arabic handwriting recognition using Hid- den Markov Models", Proceeding of IWFHR'06, 10th International Workshop on Frontiers in Handwriting Recognition, pp. 85-90, La Baule, France 2006.

[37] G.Gaillat, Méthodes syntaxiques de la reconnaissance de formes, ENSTA, Paris, 1983.

[38] J.Kittler, M.Hatef R.Duin & J.Matas, "On combining classi ers", IEEE on pattern analysis and machine intelligence, vol20, n°3, pp 226-239, 1998.

[39] S.AGNE, and M. ROGGER, Benchmarking of Document Page Segmentation , Part of the IS&T/SPIE Conference on Document Recognition and Retrieval VII, San Jose, California, p. 165-171, January 2000.

[40] D.GACEB Contributions Au TriAautomatique De Documents Et De Courrier D'en- treprises ,INSA De Lyon,Thèse, 2009.

[41] Yao-Hong Tsai, A New Approach for Image Thresholding under Uneven Lighting Conditions, Computer and Information Science, ICIS 2007. 6th IEEE/ACIS Interna- tional Conference on, 11-13, July 2007. pp.123-127.

[42] N. Babaguchi and al, Connectionist model binarisation, Pattern Recognition, ICPR 1990. Proceedings, 10th International Conference, 16-21 June 1990, vol.2, pp. 51-56.

[44] A New method for gray-level picture threshold using the entropy of the histogram, Computer Vision, Graphics, and Image Processing, 29, p. 273-285, 1985.

[45] J. BERNSEN. Dynamic thresholding of grey-level images. in Proc. 8th International Conference on Pattern Recognition, p.

[46] A.J.O. TRIER. Goal-directed evaluation of binarization methods, IEEE Transactions on Pattern Analysis and Machine Intelligence, 17(12), p. 1191-1201, 1995.

[47] W. NIBLACK. An Introduction to Digital Image Processing, Ed. Prentice Hall,

Englewood Clis pp 115-116, 1986.

[48] J. HE, Q. D. M. DO, A. C. DOWNTON, J. H. KIM. A comparison of binarization methods for historical archive documents. International Conference on Document Analysis and Recognition (ICDAR), p. 538-542, 2005.

[49] Y.Sculo Introduction au traitement d'images Détection de contours et segmenta- tion ,2009.

[50] M. Bergounioux Quelques méthodes de ltrage en Traitement d'Image ,article, Jan 2011.
[51] R.Smith An Overview of the Tesseract OCR Engine ,article,2006.

[52] K. KHURSHID, I. SIDDIQI, C. FAURE, N. VINCENT. Comparison of Niblack inspired Binarization methods for ancient documents. 16th International conference on Document Recognition and Retrieval, USA , 2009.

[53] E. Zemouri, Y. Chibani, and Y. Brik "Restoration based Contourlet Transform for historical document image binarization." ,Multimedia Computing and Systems (ICMCS), 2014 International Conference on IEEE, 2014.

[54] D.Gaceb , F.Lebourgeois, and J. Duong. "Adaptative Smart-Binarization Method : For Images of Business Documents." Document Analysis and Recognition (ICDAR), 2013 12th International Conference on. IEEE, 2013.

[55] V.Rabeux, N.Journet, A.Vialard et J-P.Domenger. "Quality evaluation of ancient digitized documents for binarization prediction." Document Analysis and Recognition (ICDAR), 2013 12th International Conference on. IEEE, 2013.

[56] M.Wagdy, I.Faye, and D.Rohaya. "Fast and e cient document image clean up and binarization based on retinex theory." Signal Processing and its Applications (CSPA), 2013 IEEE 9th International Colloquium on. IEEE, 2013.

[57] B.Smith,H.Elisa , and C.An. "E ect of" ground truth" on image binarization." Do- cument Analysis Systems (DAS), 2012 10th IAPR International Workshop on. IEEE, 2012.

[58] A. H. Ahmed, M. Afifi, M. Korashy, E. K.William, M. A. El-sattar, and Z. Hafez, "OCR System for Poor Quality Images Using ChainCode Representation," The 1st International Conference on Advanced Intelligent System and Informatics (AISI2015). Beni Suef, Egypt. Springer, Cham, pp. 151-161, 2016.

[59] C. Patel, A. Patel, and D. Patel, "Optical Character Recognition by Open Source OCR Tool Tesseract: A Case Study," International Journal of Computer Applications, vol. 55, no. 10, pp. 50-56, 2012.

[60] A. M. A. M. Asif, S. A. Hannan, Y. Perwej, and M. A. Vithalrao, "An Overview And Applications Of Optical Character Recognition," International Journal of Advance Research In Science And Engineering, vol. 3, no. 7, pp. 261-274, 2014.

[61] AnandMishra, K. Alahari, and C. V. Jawahar, "Top-Down and Bottom-up Cues for Scene Text Recognition," Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, pp. 2687- 2694, 2012.

[62] L. Neumann, and J. Matas, "A method for text localization and recognition in real-world images," Asian Conference on Computer Vision. Springer, Berlin, Heidelberg, pp. 770-783, 2010.

[63] A. Coates, B. Carpenter, C. Case, S. Satheesh, B. Suresh, T. Wang, D. J. Wu, and A. Y. Ng, "Text Detection and Character Recognition in Scene Images with Unsupervised Feature Learning," International Conference on Document Analysis and Recognition (ICDAR). IEEE, pp. 440-445, 2011.

[64] S. Babu, Z. A. Masood, S. Munir, S. Adnan, and I. Bari, "Android Based Optical Character Recognition for Noisy Document Images," International Journal of Computer Science and Information Security (IJCSIS), vol. 14, no. 1, pp. 34-37, 2016.

[64] T. E. d. Campos, B. R. Babu, and M. Varma, "Character Recognition In Natural Images," 2009.

[65] M. A. Mohamad, D. Nasien, H. Hassan, and H. Haron, "A Review on Feature Extraction and Feature Selection for Handwritten Character Recognition," International Journal of Advanced Computer Science and Applications (IJACSA), vol. 6, no. 2, pp. 204

[66] : Wenting LU, Honggang ZHANG, Kunyan LAN, Jun GUO, Size-self-adaptive recognition method of vehicle manufacturer logos based on feature extraction and SVM classifier. Higher Education Press and Springer-Verlag 2009.

[68] : Miriam Butzke, Alexandre G. Silva, Marcelo da S. Hounsell, Mauricio A. Pillon. Automatic Recognition of Vehicle - Attributes Color Classification and Logo Segmentation. Departamento de Ciencia da Computacao, Universidade do Estado de Santa Catarina, BRAZIL

[69] : Wenju Li and Ling Li. A Novel Approach for Vehicle-logo Location Based on Edge Detection and Morphological Filter. School of Computer and Information Technology Liaoning normal university Dalian , China

[70] : WANG Yunqiong ,LIU Zhifang , XIAO Fei. Fast Coarse-to-Fine Vehicle Logo Detection and Recognition Method. International Conference on Robotics and Biomimetics, December 15 -18, 2007,

WEBgraphie

[71] https://www.anaconda.com/distribution/

[72] https://python-prepa.github.io/intro.html

[73] https://opencv.org/

[74] https://www.datacamp.com/community/tutorials/gui-tkinter-python

ANNEXE

1)Licence plaque detection:

```
# DetectChars.py
import os
import cv2
import numpy as np
import math
import random
import Main
import Preprocess
import PossibleChar
# module level variables
##########################################################################
kNearest = cv2.ml.KNearest_create()
      # constants for checkIfPossibleChar, this checks one possible char only (does not compare to
another char)
MIN_PIXEL_WIDTH = 2
MIN_PIXEL_HEIGHT = 8
MIN_ASPECT_RATIO = 0.25
MAX_ASPECT_RATIO = 1.0
MIN_PIXEL_AREA = 80
      # constants for comparing two chars
MIN_DIAG_SIZE_MULTIPLE_AWAY = 0.3
MAX_DIAG_SIZE_MULTIPLE_AWAY = 5.0
MAX_CHANGE_IN_AREA = 0.5
MAX_CHANGE_IN_WIDTH = 0.8
MAX_CHANGE_IN_HEIGHT = 0.2
MAX_ANGLE_BETWEEN_CHARS = 12.0

      # other constants
MIN_NUMBER_OF_MATCHING_CHARS = 3
RESIZED_CHAR_IMAGE_WIDTH = 20
RESIZED_CHAR_IMAGE_HEIGHT = 30
MIN_CONTOUR_AREA = 100
##########################################################################################
#################
def loadKNNDataAndTrainKNN():
   allContoursWithData = []            # declare empty lists,
   validContoursWithData = []          # we will fill these shortly
   try:
      npaClassifications = np.loadtxt("classifications.txt", np.float32)              # read in training
classifications
   except:                                                 # if file could not be opened
      print("error, unable to open classifications.txt, exiting program\n")  # show error message
      os.system("pause")
      return False                                            # and return False
   # end try
   try:
      npaFlattenedImages = np.loadtxt("flattened_images.txt", np.float32)             # read in training
images
   except:                                                 # if file could not be opened
      print("error, unable to open flattened_images.txt, exiting program\n")  # show error message
      os.system("pause")
```

```
        return False                                                # and return False
    # end try
    npaClassifications = npaClassifications.reshape((npaClassifications.size, 1))      # reshape numpy
array to 1d, necessary to pass to call to train

    kNearest.setDefaultK(1)                                         # set default K to 1
    kNearest.train(npaFlattenedImages, cv2.ml.ROW_SAMPLE, npaClassifications)         # train KNN
object
    return True                         # if we got here training was successful so return true
# end function
###################################################################################
#################
def detectCharsInPlates(listOfPossiblePlates):
    intPlateCounter = 0
    imgContours = None
    contours = []
    if len(listOfPossiblePlates) == 0:          # if list of possible plates is empty
        return listOfPossiblePlates             # return
    # end if
            # at this point we can be sure the list of possible plates has at least one plate

    for possiblePlate in listOfPossiblePlates:          # for each possible plate, this is a big for loop that
takes up most of the function

        possiblePlate.imgGrayscale, possiblePlate.imgThresh =
Preprocess.preprocess(possiblePlate.imgPlate)     # preprocess to get grayscale and threshold images

        if Main.showSteps == True: # show steps
###################################################
            cv2.imshow("5a", possiblePlate.imgPlate)
            cv2.imshow("5b", possiblePlate.imgGrayscale)
            cv2.imshow("5c", possiblePlate.imgThresh)
        # end if # show steps
#####################################################################

                # increase size of plate image for easier viewing and char detection
        possiblePlate.imgThresh = cv2.resize(possiblePlate.imgThresh, (0, 0), fx = 1.6, fy = 1.6)

                # threshold again to eliminate any gray areas
        thresholdValue, possiblePlate.imgThresh = cv2.threshold(possiblePlate.imgThresh, 0.0, 255.0,
cv2.THRESH_BINARY | cv2.THRESH_OTSU)

        if Main.showSteps == True: # show steps
###################################################
            cv2.imshow("5d", possiblePlate.imgThresh)
        # end if # show steps
#####################################################################

                # find all possible chars in the plate,
                # this function first finds all contours, then only includes contours that could be chars
(without comparison to other chars yet)
        listOfPossibleCharsInPlate = findPossibleCharsInPlate(possiblePlate.imgGrayscale,
possiblePlate.imgThresh)
```

```
    if Main.showSteps == True: # show steps
###################################################
        height, width, numChannels = possiblePlate.imgPlate.shape
        imgContours = np.zeros((height, width, 3), np.uint8)
        del contours[:]                                 # clear the contours list

        for possibleChar in listOfPossibleCharsInPlate:
            contours.append(possibleChar.contour)
        # end for

        cv2.drawContours(imgContours, contours, -1, Main.SCALAR_WHITE)

        cv2.imshow("6", imgContours)
    # end if # show steps
#####################################################################

            # given a list of all possible chars, find groups of matching chars within the plate
    listOfListsOfMatchingCharsInPlate = findListOfListsOfMatchingChars(listOfPossibleCharsInPlate)

    if Main.showSteps == True: # show steps
###################################################
        imgContours = np.zeros((height, width, 3), np.uint8)
        del contours[:]

        for listOfMatchingChars in listOfListsOfMatchingCharsInPlate:
            intRandomBlue = random.randint(0, 255)
            intRandomGreen = random.randint(0, 255)
            intRandomRed = random.randint(0, 255)

            for matchingChar in listOfMatchingChars:
                contours.append(matchingChar.contour)
            # end for
            cv2.drawContours(imgContours, contours, -1, (intRandomBlue, intRandomGreen,
intRandomRed))
        # end for
        cv2.imshow("7", imgContours)
    # end if # show steps
#####################################################################

    if (len(listOfListsOfMatchingCharsInPlate) == 0):                       # if no groups of matching
chars were found in the plate

        if Main.showSteps == True: # show steps
###################################################
            print("chars found in plate number " + str(
                intPlateCounter) + " = (none), click on any image and press a key to continue . . .")
            intPlateCounter = intPlateCounter + 1
            cv2.destroyWindow("8")
            cv2.destroyWindow("9")
            cv2.destroyWindow("10")
            cv2.waitKey(0)
        # end if # show steps
#################################################################
```

```
            possiblePlate.strChars = ""
            continue                                        # go back to top of for loop
        # end if

        for i in range(0, len(listOfListsOfMatchingCharsInPlate)):                  # within each list of
matching chars
            listOfListsOfMatchingCharsInPlate[i].sort(key = lambda matchingChar:
matchingChar.intCenterX)        # sort chars from left to right
            listOfListsOfMatchingCharsInPlate[i] =
removeInnerOverlappingChars(listOfListsOfMatchingCharsInPlate[i])           # and remove inner
overlapping chars
        # end for

        if Main.showSteps == True: # show steps
#####################################################
            imgContours = np.zeros((height, width, 3), np.uint8)

            for listOfMatchingChars in listOfListsOfMatchingCharsInPlate:
                intRandomBlue = random.randint(0, 255)
                intRandomGreen = random.randint(0, 255)
                intRandomRed = random.randint(0, 255)

                del contours[:]

                for matchingChar in listOfMatchingChars:
                    contours.append(matchingChar.contour)
                # end for

                cv2.drawContours(imgContours, contours, -1, (intRandomBlue, intRandomGreen,
intRandomRed))
            # end for
            cv2.imshow("8", imgContours)
        # end if # show steps
#####################################################################

            # within each possible plate, suppose the longest list of potential matching chars is the
actual list of chars
        intLenOfLongestListOfChars = 0
        intIndexOfLongestListOfChars = 0

            # loop through all the vectors of matching chars, get the index of the one with the most
chars
        for i in range(0, len(listOfListsOfMatchingCharsInPlate)):
            if len(listOfListsOfMatchingCharsInPlate[i]) > intLenOfLongestListOfChars:
                intLenOfLongestListOfChars = len(listOfListsOfMatchingCharsInPlate[i])
                intIndexOfLongestListOfChars = i
            # end if
        # end for

            # suppose that the longest list of matching chars within the plate is the actual list of chars
        longestListOfMatchingCharsInPlate =
listOfListsOfMatchingCharsInPlate[intIndexOfLongestListOfChars]
```

```
        if Main.showSteps == True: # show steps
#####################################################
            imgContours = np.zeros((height, width, 3), np.uint8)
            del contours[:]

            for matchingChar in longestListOfMatchingCharsInPlate:
                contours.append(matchingChar.contour)
            # end for

            cv2.drawContours(imgContours, contours, -1, Main.SCALAR_WHITE)

            cv2.imshow("9", imgContours)
        # end if # show steps
#####################################################################

        possiblePlate.strChars = recognizeCharsInPlate(possiblePlate.imgThresh,
longestListOfMatchingCharsInPlate)

        if Main.showSteps == True: # show steps
#####################################################
            print("chars found in plate number " + str(
                intPlateCounter) + " = " + possiblePlate.strChars + ", click on any image and press a key to
continue . . .")
            intPlateCounter = intPlateCounter + 1
            cv2.waitKey(0)
        # end if # show steps
#####################################################################

    # end of big for loop that takes up most of the function

    if Main.showSteps == True:
        print("\nchar detection complete, click on any image and press a key to continue . . .\n")
        cv2.waitKey(0)
    # end if

    return listOfPossiblePlates
# end function

#########################################################################################
#################
def findPossibleCharsInPlate(imgGrayscale, imgThresh):
    listOfPossibleChars = []                    # this will be the return value
    contours = []
    imgThreshCopy = imgThresh.copy()

            # find all contours in plate
    # imgContours, contours, npaHierarchy = cv2.findContours(imgThreshCopy, cv2.RETR_LIST,
cv2.CHAIN_APPROX_SIMPLE)
    contours, npaHierarchy = cv2.findContours(imgThreshCopy, cv2.RETR_LIST,
cv2.CHAIN_APPROX_SIMPLE)

    for contour in contours:                    # for each contour
        possibleChar = PossibleChar.PossibleChar(contour)
```

```
      if checkIfPossibleChar(possibleChar):            # if contour is a possible char, note this does not
compare to other chars (yet) . . .
         listOfPossibleChars.append(possibleChar)      # add to list of possible chars
      # end if
   # end if

   return listOfPossibleChars
# end function

#################################################################################################
#################
def checkIfPossibleChar(possibleChar):
         # this function is a 'first pass' that does a rough check on a contour to see if it could be a char,
         # note that we are not (yet) comparing the char to other chars to look for a group
   if (possibleChar.intBoundingRectArea > MIN_PIXEL_AREA and
      possibleChar.intBoundingRectWidth > MIN_PIXEL_WIDTH and
possibleChar.intBoundingRectHeight > MIN_PIXEL_HEIGHT and
      MIN_ASPECT_RATIO < possibleChar.fltAspectRatio and possibleChar.fltAspectRatio <
MAX_ASPECT_RATIO):
      return True
   else:
      return False
   # end if
# end function

#################################################################################################
#################
def findListOfListsOfMatchingChars(listOfPossibleChars):
         # with this function, we start off with all the possible chars in one big list
         # the purpose of this function is to re-arrange the one big list of chars into a list of lists of
matching chars,
         # note that chars that are not found to be in a group of matches do not need to be considered
further
   listOfListsOfMatchingChars = []              # this will be the return value

   for possibleChar in listOfPossibleChars:                  # for each possible char in the one big list of
chars
      listOfMatchingChars = findListOfMatchingChars(possibleChar, listOfPossibleChars)       # find all
chars in the big list that match the current char

      listOfMatchingChars.append(possibleChar)               # also add the current char to current
possible list of matching chars

      if len(listOfMatchingChars) < MIN_NUMBER_OF_MATCHING_CHARS:     # if current possible list
of matching chars is not long enough to constitute a possible plate
         continue                         # jump back to the top of the for loop and try again with next char,
note that it's not necessary
                                 # to save the list in any way since it did not have enough chars to be a
possible plate
      # end if

                                 # if we get here, the current list passed test as a "group" or "cluster" of
matching chars
```

```
        listOfListsOfMatchingChars.append(listOfMatchingChars)      # so add to our list of lists of matching chars

        listOfPossibleCharsWithCurrentMatchesRemoved = []

                                    # remove the current list of matching chars from the big list so we don't use those same chars twice,
                                    # make sure to make a new big list for this since we don't want to change the original big list
        listOfPossibleCharsWithCurrentMatchesRemoved = list(set(listOfPossibleChars) - set(listOfMatchingChars))

        recursiveListOfListsOfMatchingChars = findListOfListsOfMatchingChars(listOfPossibleCharsWithCurrentMatchesRemoved)      # recursive call

        for recursiveListOfMatchingChars in recursiveListOfListsOfMatchingChars:        # for each list of matching chars found by recursive call
            listOfListsOfMatchingChars.append(recursiveListOfMatchingChars)             # add to our original list of lists of matching chars
        # end for

        break       # exit for

    # end for

    return listOfListsOfMatchingChars
# end function

#################################################################################################
def findListOfMatchingChars(possibleChar, listOfChars):
        # the purpose of this function is, given a possible char and a big list of possible chars,
        # find all chars in the big list that are a match for the single possible char, and return those matching chars as a list
    listOfMatchingChars = []                # this will be the return value

    for possibleMatchingChar in listOfChars:                # for each char in big list
        if possibleMatchingChar == possibleChar:    # if the char we attempting to find matches for is the exact same char as the char in the big list we are currently checking
                                    # then we should not include it in the list of matches b/c that would end up double including the current char
            continue                        # so do not add to list of matches and jump back to top of for loop
        # end if
                # compute stuff to see if chars are a match
        fltDistanceBetweenChars = distanceBetweenChars(possibleChar, possibleMatchingChar)

        fltAngleBetweenChars = angleBetweenChars(possibleChar, possibleMatchingChar)

        fltChangeInArea = float(abs(possibleMatchingChar.intBoundingRectArea - possibleChar.intBoundingRectArea)) / float(possibleChar.intBoundingRectArea)

        fltChangeInWidth = float(abs(possibleMatchingChar.intBoundingRectWidth - possibleChar.intBoundingRectWidth)) / float(possibleChar.intBoundingRectWidth)
```

```
      fltChangeInHeight = float(abs(possibleMatchingChar.intBoundingRectHeight -
possibleChar.intBoundingRectHeight)) / float(possibleChar.intBoundingRectHeight)

            # check if chars match
      if (fltDistanceBetweenChars < (possibleChar.fltDiagonalSize *
MAX_DIAG_SIZE_MULTIPLE_AWAY) and
          fltAngleBetweenChars < MAX_ANGLE_BETWEEN_CHARS and
          fltChangeInArea < MAX_CHANGE_IN_AREA and
          fltChangeInWidth < MAX_CHANGE_IN_WIDTH and
          fltChangeInHeight < MAX_CHANGE_IN_HEIGHT):

          listOfMatchingChars.append(possibleMatchingChar)       # if the chars are a match, add the
current char to list of matching chars
      # end if
   # end for

   return listOfMatchingChars              # return result
# end function

###################################################################################################
# use Pythagorean theorem to calculate distance between two chars
def distanceBetweenChars(firstChar, secondChar):
   intX = abs(firstChar.intCenterX - secondChar.intCenterX)
   intY = abs(firstChar.intCenterY - secondChar.intCenterY)

   return math.sqrt((intX ** 2) + (intY ** 2))
# end function

###################################################################################################
# use basic trigonometry (SOH CAH TOA) to calculate angle between chars
def angleBetweenChars(firstChar, secondChar):
   fltAdj = float(abs(firstChar.intCenterX - secondChar.intCenterX))
   fltOpp = float(abs(firstChar.intCenterY - secondChar.intCenterY))

   if fltAdj != 0.0:                       # check to make sure we do not divide by zero if the center X
positions are equal, float division by zero will cause a crash in Python
      fltAngleInRad = math.atan(fltOpp / fltAdj)      # if adjacent is not zero, calculate angle
   else:
      fltAngleInRad = 1.5708                      # if adjacent is zero, use this as the angle, this is to be
consistent with the C++ version of this program
   # end if

   fltAngleInDeg = fltAngleInRad * (180.0 / math.pi)       # calculate angle in degrees

   return fltAngleInDeg
# end function

###################################################################################################
# if we have two chars overlapping or to close to each other to possibly be separate chars, remove
the inner (smaller) char,
# this is to prevent including the same char twice if two contours are found for the same char,
```

```
# for example for the letter 'O' both the inner ring and the outer ring may be found as contours, but we should only include the char once
def removeInnerOverlappingChars(listOfMatchingChars):
   listOfMatchingCharsWithInnerCharRemoved = list(listOfMatchingChars)              # this will be the return value

   for currentChar in listOfMatchingChars:
      for otherChar in listOfMatchingChars:
         if currentChar != otherChar:       # if current char and other char are not the same char . . .
                                            # if current char and other char have center points at almost the same location . . .
            if distanceBetweenChars(currentChar, otherChar) < (currentChar.fltDiagonalSize * MIN_DIAG_SIZE_MULTIPLE_AWAY):
                        # if we get in here we have found overlapping chars
                        # next we identify which char is smaller, then if that char was not already removed on a previous pass, remove it
               if currentChar.intBoundingRectArea < otherChar.intBoundingRectArea:        # if current char is smaller than other char
                  if currentChar in listOfMatchingCharsWithInnerCharRemoved:            # if current char was not already removed on a previous pass . . .
                     listOfMatchingCharsWithInnerCharRemoved.remove(currentChar)        # then remove current char
                  # end if
               else:                                                # else if other char is smaller than current char
                  if otherChar in listOfMatchingCharsWithInnerCharRemoved:              # if other char was not already removed on a previous pass . . .
                     listOfMatchingCharsWithInnerCharRemoved.remove(otherChar)          # then remove other char
                  # end if
               # end if
            # end if
         # end if
      # end for
   # end for
   return listOfMatchingCharsWithInnerCharRemoved
# end function

#################################################################################################
# this is where we apply the actual char recognition
def recognizeCharsInPlate(imgThresh, listOfMatchingChars):
   strChars = ""              # this will be the return value, the chars in the lic plate
   height, width = imgThresh.shape
   imgThreshColor = np.zeros((height, width, 3), np.uint8)
   listOfMatchingChars.sort(key = lambda matchingChar: matchingChar.intCenterX)       # sort chars from left to right
   cv2.cvtColor(imgThresh, cv2.COLOR_GRAY2BGR, imgThreshColor)                # make color version of threshold image so we can draw contours in color on it
   for currentChar in listOfMatchingChars:                                  # for each char in plate
      pt1 = (currentChar.intBoundingRectX, currentChar.intBoundingRectY)
      pt2 = ((currentChar.intBoundingRectX + currentChar.intBoundingRectWidth), (currentChar.intBoundingRectY + currentChar.intBoundingRectHeight))
```

```
      cv2.rectangle(imgThreshColor, pt1, pt2, Main.SCALAR_GREEN, 2)          # draw green box around
the char
            # crop char out of threshold image
      imgROI = imgThresh[currentChar.intBoundingRectY : currentChar.intBoundingRectY +
currentChar.intBoundingRectHeight,
                  currentChar.intBoundingRectX : currentChar.intBoundingRectX +
currentChar.intBoundingRectWidth]
      imgROIResized = cv2.resize(imgROI, (RESIZED_CHAR_IMAGE_WIDTH,
RESIZED_CHAR_IMAGE_HEIGHT))          # resize image, this is necessary for char recognition
      npaROIResized = imgROIResized.reshape((1, RESIZED_CHAR_IMAGE_WIDTH *
RESIZED_CHAR_IMAGE_HEIGHT))       # flatten image into 1d numpy array
      npaROIResized = np.float32(npaROIResized)              # convert from 1d numpy array of ints to 1d
numpy array of floats
      retval, npaResults, neigh_resp, dists = kNearest.findNearest(npaROIResized, k = 1)            #
finally we can call findNearest !!!
      strCurrentChar = str(chr(int(npaResults[0][0])))          # get character from results
      strChars = strChars + strCurrentChar                     # append current char to full string
   # end for
   if Main.showSteps == True: # show steps
#######################################################
      cv2.imshow("10", imgThreshColor)
   # end if # show steps
#########################################################################
   return strChars
# end function
```

2) License plate recognition:

```
# DetectPlates.py
import cv2
import numpy as np
import math
import Main
import random
import Preprocess
import DetectChars
import PossiblePlate
import PossibleChar
# module level variables
##########################################################################
PLATE_WIDTH_PADDING_FACTOR = 1.3
PLATE_HEIGHT_PADDING_FACTOR = 1.5
##########################################################################################
#################
def detectPlatesInScene(imgOriginalScene):
   listOfPossiblePlates = []                   # this will be the return value
   height, width, numChannels = imgOriginalScene.shape
   imgGrayscaleScene = np.zeros((height, width, 1), np.uint8)
   imgThreshScene = np.zeros((height, width, 1), np.uint8)
   imgContours = np.zeros((height, width, 3), np.uint8)
   cv2.destroyAllWindows()
   if Main.showSteps == True: # show steps
#########################################################
      cv2.imshow("0", imgOriginalScene)
   # end if # show steps
##########################################################################
   imgGrayscaleScene, imgThreshScene = Preprocess.preprocess(imgOriginalScene)        # preprocess
to get grayscale and threshold images
   if Main.showSteps == True: # show steps
#########################################################
      cv2.imshow("1a", imgGrayscaleScene)
      cv2.imshow("1b", imgThreshScene)
   # end if # show steps
##########################################################################
         # find all possible chars in the scene,
         # this function first finds all contours, then only includes contours that could be chars (without
comparison to other chars yet)
   listOfPossibleCharsInScene = findPossibleCharsInScene(imgThreshScene)
   if Main.showSteps == True: # show steps
#########################################################
      print("step 2 - len(listOfPossibleCharsInScene) = " + str(
         len(listOfPossibleCharsInScene)))  # 131 with MCLRNF1 image
      imgContours = np.zeros((height, width, 3), np.uint8)
      contours = []
      for possibleChar in listOfPossibleCharsInScene:
         contours.append(possibleChar.contour)       # end for
      cv2.drawContours(imgContours, contours, -1, Main.SCALAR_WHITE)
      cv2.imshow("2b", imgContours)    # end if # show steps
##########################################################################
         # given a list of all possible chars, find groups of matching chars
         # in the next steps each group of matching chars will attempt to be recognized as a plate
```

```
    listOfListsOfMatchingCharsInScene =
DetectChars.findListOfListsOfMatchingChars(listOfPossibleCharsInScene)
    if Main.showSteps == True: # show steps
###################################################
        print("step 3 - listOfListsOfMatchingCharsInScene.Count = " + str(
            len(listOfListsOfMatchingCharsInScene)))  # 13 with MCLRNF1 image
        imgContours = np.zeros((height, width, 3), np.uint8)
        for listOfMatchingChars in listOfListsOfMatchingCharsInScene:
            intRandomBlue = random.randint(0, 255)
            intRandomGreen = random.randint(0, 255)
            intRandomRed = random.randint(0, 255)
            contours = []
            for matchingChar in listOfMatchingChars:
                contours.append(matchingChar.contour)
            # end for
            cv2.drawContours(imgContours, contours, -1, (intRandomBlue, intRandomGreen,
intRandomRed))       # end for
        cv2.imshow("3", imgContours)
    # end if # show steps
#########################################################################
    for listOfMatchingChars in listOfListsOfMatchingCharsInScene:               # for each group of
matching chars
        possiblePlate = extractPlate(imgOriginalScene, listOfMatchingChars)        # attempt to extract
plate
        if possiblePlate.imgPlate is not None:                          # if plate was found
            listOfPossiblePlates.append(possiblePlate)                # add to list of possible plates
        # end if
    # end for
    print("\n" + str(len(listOfPossiblePlates)) + " possible plates found")  # 13 with MCLRNF1 image
    if Main.showSteps == True: # show steps
###################################################
        print("\n")
        cv2.imshow("4a", imgContours)
    for i in range(0, len(listOfPossiblePlates)):
            p2fRectPoints = cv2.boxPoints(listOfPossiblePlates[i].rrLocationOfPlateInScene)
            cv2.line(imgContours, tuple(p2fRectPoints[0]), tuple(p2fRectPoints[1]), Main.SCALAR_RED, 2)
            cv2.line(imgContours, tuple(p2fRectPoints[1]), tuple(p2fRectPoints[2]), Main.SCALAR_RED, 2)
            cv2.line(imgContours, tuple(p2fRectPoints[2]), tuple(p2fRectPoints[3]), Main.SCALAR_RED, 2)
            cv2.line(imgContours, tuple(p2fRectPoints[3]), tuple(p2fRectPoints[0]), Main.SCALAR_RED, 2)
            cv2.imshow("4a", imgContours)
            print("possible plate " + str(i) + ", click on any image and press a key to continue . . .")
           cv2.imshow("4b", listOfPossiblePlates[i].imgPlate)
            cv2.waitKey(0)
        # end for
        print("\nplate detection complete, click on any image and press a key to begin char recognition .
. .\n")
        cv2.waitKey(0)
    # end if # show steps
#########################################################################
    return listOfPossiblePlates
# end function
#########################################################################################
#################
def findPossibleCharsInScene(imgThresh):
```

```
    listOfPossibleChars = []                # this will be the return value
    intCountOfPossibleChars = 0
    imgThreshCopy = imgThresh.copy()
   # imgContours, contours, npaHierarchy = cv2.findContours(imgThreshCopy, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)   # find all contours
    contours, npaHierarchy = cv2.findContours(imgThreshCopy, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
    height, width = imgThresh.shape
    imgContours = np.zeros((height, width, 3), np.uint8)
    for i in range(0, len(contours)):                    # for each contour
        if Main.showSteps == True: # show steps
#######################################################
            cv2.drawContours(imgContours, contours, i, Main.SCALAR_WHITE)
        # end if # show steps
#########################################################################
        possibleChar = PossibleChar.PossibleChar(contours[i])
        if DetectChars.checkIfPossibleChar(possibleChar):                # if contour is a possible char, note
this does not compare to other chars (yet) . . .
            intCountOfPossibleChars = intCountOfPossibleChars + 1          # increment count of possible
chars
            listOfPossibleChars.append(possibleChar)                    # and add to list of possible chars
        # end if
    # end for
    if Main.showSteps == True: # show steps
#######################################################
        print("\nstep 2 - len(contours) = " + str(len(contours)))  # 2362 with MCLRNF1 image
        print("step 2 - intCountOfPossibleChars = " + str(intCountOfPossibleChars))  # 131 with MCLRNF1
image
        cv2.imshow("2a", imgContours)
    # end if # show steps
#########################################################################
    return listOfPossibleChars
# end function
#################################################################################################
#################
def extractPlate(imgOriginal, listOfMatchingChars):
    possiblePlate = PossiblePlate.PossiblePlate()          # this will be the return value
    listOfMatchingChars.sort(key = lambda matchingChar: matchingChar.intCenterX)       # sort chars
from left to right based on x position
            # calculate the center point of the plate
    fltPlateCenterX = (listOfMatchingChars[0].intCenterX +
listOfMatchingChars[len(listOfMatchingChars) - 1].intCenterX) / 2.0
    fltPlateCenterY = (listOfMatchingChars[0].intCenterY +
listOfMatchingChars[len(listOfMatchingChars) - 1].intCenterY) / 2.0
    ptPlateCenter = fltPlateCenterX, fltPlateCenterY
            # calculate plate width and height
    intPlateWidth = int((listOfMatchingChars[len(listOfMatchingChars) - 1].intBoundingRectX +
listOfMatchingChars[len(listOfMatchingChars) - 1].intBoundingRectWidth -
listOfMatchingChars[0].intBoundingRectX) * PLATE_WIDTH_PADDING_FACTOR)
    intTotalOfCharHeights = 0
    for matchingChar in listOfMatchingChars:
        intTotalOfCharHeights = intTotalOfCharHeights + matchingChar.intBoundingRectHeight
    # end for
    fltAverageCharHeight = intTotalOfCharHeights / len(listOfMatchingChars)
```

```
    intPlateHeight = int(fltAverageCharHeight * PLATE_HEIGHT_PADDING_FACTOR)
        # calculate correction angle of plate region
    fltOpposite = listOfMatchingChars[len(listOfMatchingChars) - 1].intCenterY -
listOfMatchingChars[0].intCenterY
    fltHypotenuse = DetectChars.distanceBetweenChars(listOfMatchingChars[0],
listOfMatchingChars[len(listOfMatchingChars) - 1])
    fltCorrectionAngleInRad = math.asin(fltOpposite / fltHypotenuse)
    fltCorrectionAngleInDeg = fltCorrectionAngleInRad * (180.0 / math.pi)
        # pack plate region center point, width and height, and correction angle into rotated rect
member variable of plate
    possiblePlate.rrLocationOfPlateInScene = ( tuple(ptPlateCenter), (intPlateWidth, intPlateHeight),
fltCorrectionAngleInDeg )
        # final steps are to perform the actual rotation
        # get the rotation matrix for our calculated correction angle
    rotationMatrix = cv2.getRotationMatrix2D(tuple(ptPlateCenter), fltCorrectionAngleInDeg, 1.0)
    height, width, numChannels = imgOriginal.shape      # unpack original image width and height
    imgRotated = cv2.warpAffine(imgOriginal, rotationMatrix, (width, height))       # rotate the entire
image
    imgCropped = cv2.getRectSubPix(imgRotated, (intPlateWidth, intPlateHeight),
tuple(ptPlateCenter))
    possiblePlate.imgPlate = imgCropped         # copy the cropped plate image into the applicable
member variable of the possible plate
    return possiblePlate
# end function
```