

Dédicaces

Je dédie ce travail à ma famille avec tous mes sentiments de respect, de gratitude et de reconnaissance pour tous les sacrifices déployés

Je tiens à remercier toutes les personnes qui ont contribué au succès de mon stage et qui m'ont aidé lors de la rédaction de ce rapport

Safi Amine

Dédicaces

Je dédie ce travail à ma famille, mes amis, mes amours ainsi que tous ceux qui m'ont aidé pour achever ce rapport

Fakhfakh Sami

Remerciements

Nous exprimons nos profondes gratitudee et respectueuse reconnaissance à
notre encadrante Mme. Farah FOURATI

Pour sa bonne volonté d'accepter de nous encadrer, pour tout le temps qu'elle
nous a octroyée et pour tous les conseils qu'elle nous a prodiguée.

Nous remercions aussi notre encadreur de la société

M. Mohamed Amine RANNEN

Pour ses directives précieuses, et pour la qualité de ses suivis durant toute la
période de notre projet.

Nous tenons à remercier Pr. Slim CHAOUI qui a bien voulu nous faire
l'honneur de présider le jury.

Nous remercions sincèrement M. Slim BOUAZIZ d'avoir accepté d'être
l'examineur de ce manuscrit.

Nos vifs remerciements s'adressent également à nos enseignants et à nos amis,
pour leur présence chaleureuse et leurs encouragements.

Table des matières

Introduction générale	1
Chapitre 1 : Etude Préalable	2
I. Introduction	2
II. Présentation de la société d'accueil	2
III. Cahier de charges	2
IV. Les besoins attendus de l'application	3
IV.1 Les besoins non fonctionnels	3
IV.2 Les besoins fonctionnels	3
V. Diagramme UML	4
V.1 Diagramme de cas d'utilisation	4
V.2 Diagramme de classes	5
VI. Etude de l'existant	5
VII. Solution proposée	7
VIII. Planning prévisionnel	8
IX. Conclusion	9
Chapitre 2 : Reconnaissance et extraction de la plaque d'immatriculation	10
I. Introduction	11
II. Apprentissage supervisé basé sur KNN	11
III. La bibliothèque OpenCV	12
IV. Analyse de l'API existante	13
IV.1 Etude de l'API existante	13
IV.1.1 Présentation	13
IV.1.2 Fonctionnement	14
IV.2 Etude critique	17
V. Contribution	17

V.1	Apprentissage du système de classification	17
V.2	Utilisation de l'API	17
VI.	Processus de fonctionnement.....	19
VII.	Les Wrappers.....	25
VII.1	Exemple illustratif.....	27
VIII.	Conclusion.....	30
Chapitre 3 : Traitement vidéo.....		31
I.	Introduction	32
II.	Les trois approches de détection d'un véhicule.....	32
II.1	Position du véhicule connu à l'avance	32
II.2	Détecter et extraire tous les véhicules probables	33
II.2.1	Algorithme de Haarcascade	34
II.2.2	Algorithme de HOG Descriptor	35
II.2.3	Différences entre Haarcascade et HOG Descriptor	35
II.3	Détection des éventuelles plaques d'immatriculation	35
III.	Détection de la plaque d'immatriculation	37
III.1	Fonctionnement.....	37
III.2	Exemple illustratif.....	37
IV.	Conclusion.....	47
Chapitre 4 : Rétro-ingénierie et présentation de l'application.....		48
I.	Introduction	49
II.	Architecture logicielle	49
III.	Diagramme de classes par rétro-ingénierie.....	49
IV.	Présentation de l'application	54
V.	Conclusion	55
Conclusion et perspectives		56
Bibliographie		

Liste des figures

Figure 1.Scénario parfait du système	2
Figure 2.Diagramme de cas d'utilisations.....	4
Figure 3.Diagramme de classes.....	5
Figure 4.Caméra de surveillance du trafic	5
Figure 5.Plaques d'immatriculation standard utilisées dans les pays étranger	6
Figure 6.Plaques d'immatriculation tunisiennes	7
Figure 7.Fonctionnement du système	8
Figure 8.Arbre de décision de l'algorithme KNN	12
Figure 9.L'entrée du programme	13
Figure 10.Utilisation de Main.py.....	14
Figure 11.Les E/S de l'API d'origine avec un véhicule étranger	15
Figure 12.Fonctionnement de l'API.....	16
Figure 13.Extrait du dictionnaire d'apprentissage.....	17
Figure 14.Fichiers de classification	17
Figure 15.Utilisation de l'API résultante	17
Figure 16.Exécution de l'API d'origine avec une plaque d'immatriculation tunisienne...	18
Figure 17.Résultat fournie en sortie contenant le texte de la plaque d'immatriculation.	25
Figure 18.Java Wrapper implémenté.....	26
Figure 19.La classe PythonResult	26
Figure 20.Utilisation de l'API modifiée	27
Figure 21.Image d'entrée	27
Figure 22.Image du véhicule fournie en sortie	28
Figure 23.Image de la plaque fournie en sortie	28
Figure 24.Position prévue pour le passage du véhicule	32
Figure 25.Un véhicule qui passe comme prévu	32
Figure 26.Dénombrement des véhicules.....	34
Figure 27.Apprentissage de la base requise par Haarcascade.....	35
Figure 28.Les matricules encadrées.....	36
Figure 29.Code source de l'approche de détection de plaques utilisée	37

Figure 30.Onglet mode vidéo.....	50
Figure 31.Onglet mode image.....	51
Figure 32.Onglet configuration	52
Figure 33.Diagramme de classes par rétro-ingénierie	53
Figure 34.Hiérarchie du projet.....	54

Liste des tableaux

Tableau 1.Chronogramme des étapes de la réalisation	8
Tableau 2.Tableau comparatif en temps d'exécution	12
Tableau 3.Tournage du Wrapper.....	29
Tableau 4.Tableau comparatif.....	33

Introduction générale

Introduction générale

L'accroissement des réseaux routiers urbains et nationaux contemporains au cours des trois dernières décennies a fait ressortir la nécessité d'un suivi et d'une gestion efficaces de la circulation routière. Les techniques conventionnelles pour les mesures de la circulation, telles que les boucles inductives et les capteurs, souffrent de graves défauts, sont coûteux à installer, demandent des perturbations de la circulation pendant l'installation ou l'entretien et ils sont volumineux.

Quant aux systèmes basés sur la vidéo sont faciles à installer, utilisent l'infrastructure existante de surveillance du trafic. En outre, ils peuvent être facilement mis à niveau et ils offrent la souplesse nécessaire pour redessiner le système et sa fonctionnalité en changeant simplement les algorithmes du système. Ces systèmes permettent de mesurer la vitesse du véhicule, compte tenu du nombre de véhicules, du classement des véhicules et de l'identification des incidents de circulation (accidents ou encombrements importants). Il existe une grande variété de systèmes basés sur le traitement vidéo et l'image utilisant différentes méthodes pour détecter les véhicules. C'est dans ce cadre que s'inscrit notre projet de fin d'étude, nous proposons une application qui peut détecter des véhicules et extraire leurs numéros d'immatriculation.

Ce mémoire comporte quatre chapitres. Le premier chapitre présente l'entreprise d'accueil et l'étude préalable. Le second chapitre explique le mécanisme d'extraction du texte d'une plaque à partir d'une image d'un véhicule. Le troisième chapitre cite les différentes approches de détection de véhicules à partir d'une vidéo et justifie notre choix de la méthode utilisée. Le dernier chapitre est consacré à la modélisation conceptuelle, l'architecture logicielle adoptée, ainsi qu'une présentation de l'application. Enfin, nous clôturons notre rapport avec une conclusion générale et les éventuelles perspectives de ce travail.

Chapitre 1 :

Etude Préalable

I. Introduction

Ce chapitre sera réservé pour présenter l'étude préalable qui constitue une étape préliminaire pour la réalisation de notre application. Nous commençons par la présentation de l'organisme d'accueil « NextPlus », la société au sein de laquelle nous avons effectué notre projet de fin d'études. Ensuite, nous définissons le champ de l'étude et les objectifs à atteindre et analysons une solution existante sur le marché. L'analyse et le critique de l'existant nous ont permis de cerner nos objectifs afin de développer un système de qualité. Enfin, nous proposons la solution aux problèmes soulevés.

II. Présentation de la société d'accueil

Nous avons réalisé notre stage de fin d'étude de la licence fondamentale en informatique au sein de la société Next Plus. La société Next Plus, créée en février 2016, est spécialisée dans le développement informatique : applications commerciales, de paie, de comptabilité, de point de vente, d'audit, de conseil et de conception graphique. Elle s'intéresse actuellement à concevoir un système de reconnaissance de caractères.

III. Cahier de charges

Le système de surveillance du trafic est un sujet de recherche actif dans la vision par ordinateur qui tente de détecter, de reconnaître et de suivre les véhicules sur une séquence d'images. Un système de surveillance typique consiste en un réseau de caméras de trafic, qui traite la vidéo capturée et transmet les paramètres extraits.

Dans ce rapport, nous présentons le système complet de détection de véhicules, de traçage et de reconnaissance de plaque d'immatriculation. Le processus de déroulement d'un tel système se compose des étapes présentées à la figure 1.

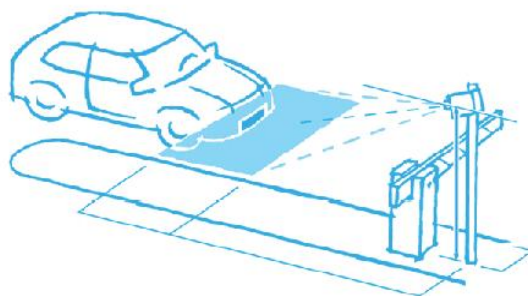


Figure 1.Scénario parfait du système

La première étape consiste à faire l'acquisition d'une image, la deuxième concerne la détection de la plaque et la dernière est consacrée à l'extraction de caractères.

IV. Les besoins attendus de l'application

Dans cette section, nous allons présenter les besoins attendus de notre application qui peuvent se diviser en deux : les besoins fonctionnels et les besoins non fonctionnels.

IV.1 Les besoins non fonctionnels

Un besoin non fonctionnel spécifie les propriétés du système, telles que les contraintes liées à l'environnement et à l'implémentation, et les exigences en matière de performances, de dépendances de plate-forme, de facilité de maintenance, d'extensibilité et de fiabilité. Les besoins non fonctionnels sont indispensables et permettent l'amélioration de la qualité logicielle de notre système. Ils agissent comme des contraintes sur les solutions, mais leur prise en considération fait éviter plusieurs incohérences dans le système. Ce dernier doit répondre aux exigences suivantes :

Ergonomie : L'application doit offrir une interface conviviale et ergonomique exploitable par l'utilisateur en envisageant toutes les interactions possibles à l'écran.

Réutilisabilité : on doit mettre en place une application évolutive et ouverte à des nouveaux modules. Pour cela l'architecture trois-tiers est la plus adaptée pour notre application.

Performance :

Temps de réponse – le chargement de l'application, ouverture d'écran et des délais de rafraîchissement.

En temps de traitement – fonctions, calculs, importations/exportations de données.

IV.2 Les besoins fonctionnels

Un besoin fonctionnel spécifie ce qu'un système doit être capable d'effectuer sans considérer les contraintes physiques. Dans le cadre de notre application, nous avons identifié:

Deux modes d'utilisation : Ce système va être utilisé par un utilisateur qui peut basculer entre deux modes : mode image et mode vidéo.

La gestion des véhicules : L'utilisateur aura la possibilité d'identifier, ajouter, modifier et supprimer des véhicules.

Reconnaissance des plaques d'immatriculation: L'application doit reconnaître les plaques d'immatriculation.

La gestion des fichiers logs : L'utilisateur de l'application doit pouvoir consulter et exporter l'historique des véhicules et des plaques d'immatriculation des véhicules détectés.

V. Diagramme UML

Dans cette section, nous présentons le diagramme de cas d'utilisations. En effet les cas d'utilisations constituent un outil précieux pour comprendre les exigences fonctionnelles d'un système.

V.1 Diagramme de cas d'utilisation

La figure 2 montre le diagramme de cas d'utilisation extrait du cahier de charges.

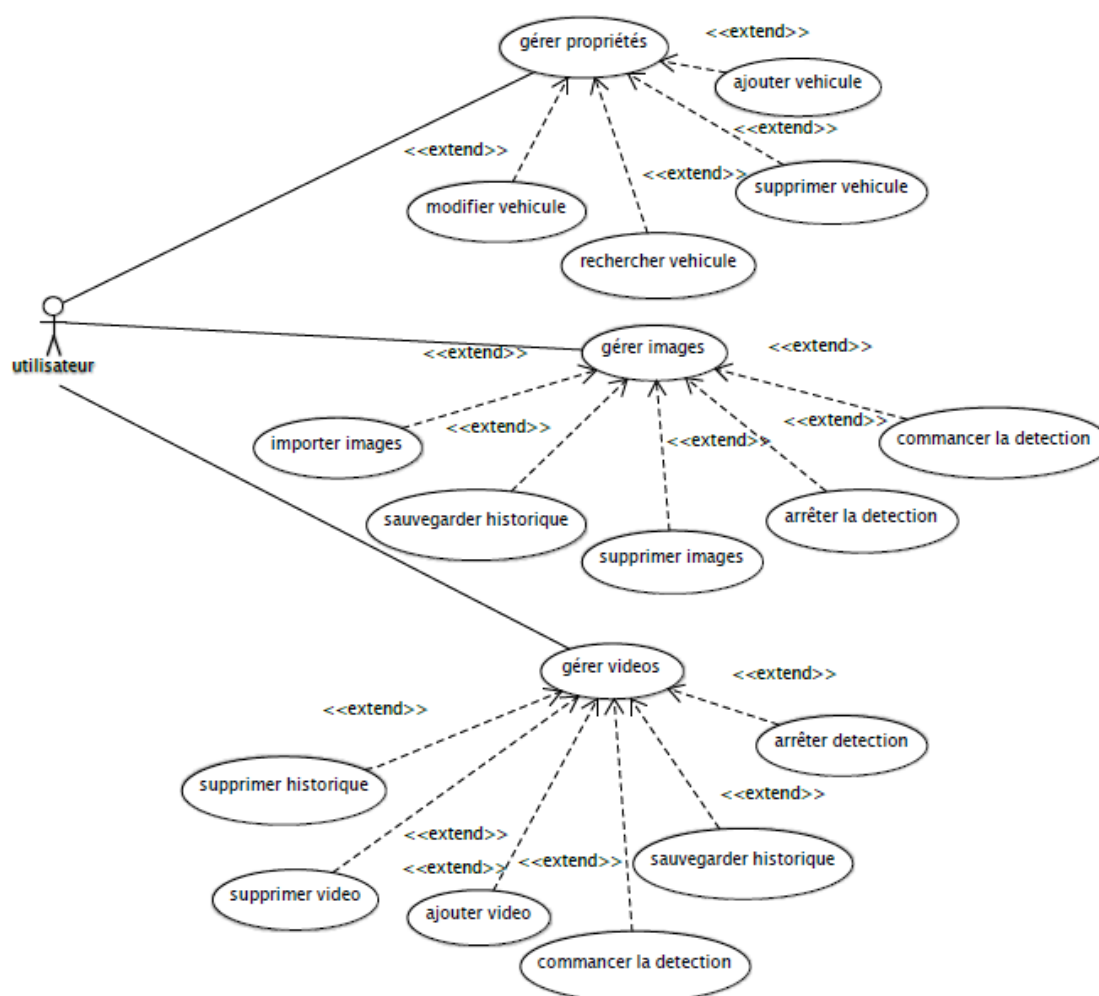


Figure 2. Diagramme de cas d'utilisations

V.2 Diagramme de classes

La figure 3 montre le diagramme de classes sur lequel nous avons raisonnés.

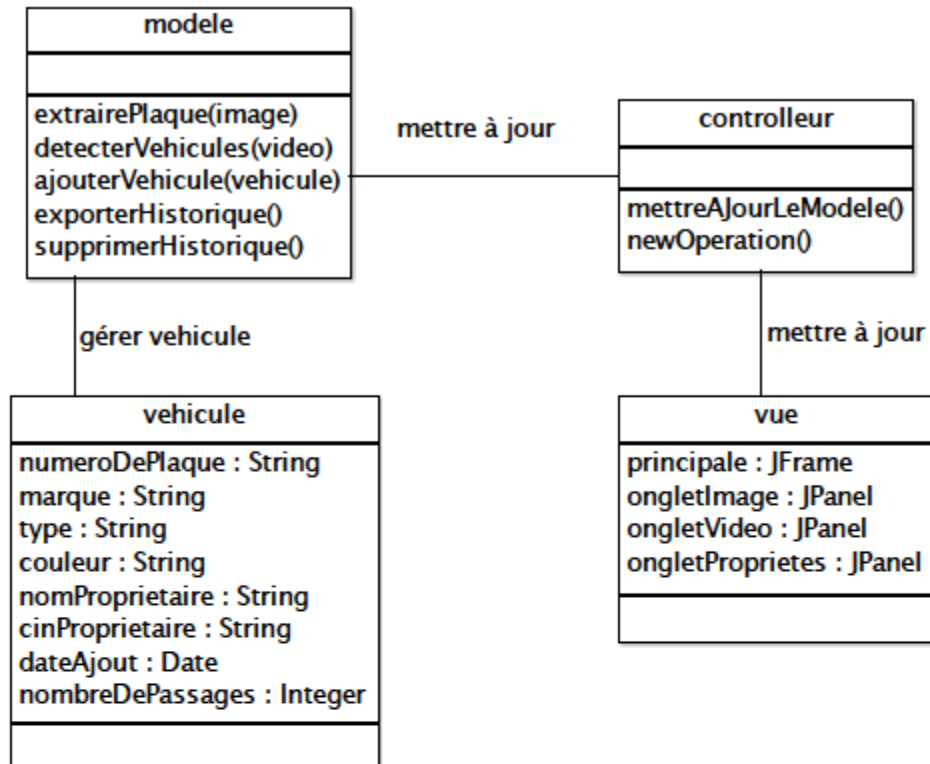


Figure 3. Diagramme de classes

VI. Etude de l'existant

L'architecture des systèmes de surveillance du trafic existants est complexe et nécessite un matériel spécifique.

(Voir Figure 4)

L'amélioration de ces systèmes doit prendre en considération plusieurs facteurs parmi eux nous citons :

- Matériel peu coûteux et non intrusif.
- Installation aisée.
- Faible entretien.
- Moins de travail civil.
- 24 heures sur 24, 7 jours sur 7.
- En conditions de nuit, quels que soient les phares.
- Reconnaissance des plaques d'immatriculation imprimées en plastique ou en métal.



Figure 4. Caméra de surveillance du trafic

Ce projet vise à concevoir et à développer un système de détection et de reconnaissance des plaques d'immatriculation qui fonctionne efficacement dans les conditions des objets en mouvement lent, robuste contre les changements d'éclairage progressif ou soudain, occlusions, temps d'identification du système devrait être aussi court que possible.

Le système doit détecter tous les types de véhicules, reconnaître toutes les plaques d'immatriculation du pays et être également résistant à tout type de perturbations, ces dernières pouvant se produire dans des images et des dommages à la plaque mécanique.

Les attributs des plaques d'immatriculation jouent un rôle important dans le processus de reconnaissance. La taille de la plaque, sa couleur, la taille de ses caractères, leurs polices, leurs couleurs, l'espacement entre eux.

La hauteur et la largeur des caractères sont strictement maintenues dans les pays développés. Il existe plusieurs pays dans le monde qui adaptent cette méthode de standardisation de la plaque d'immatriculation. Certaines des plaques d'immatriculation standard utilisée dans les pays développés sont présentées à la figure 5.



Figure 5. Plaques d'immatriculation standard utilisées dans les pays étranger

Notre objectif étant de détecter les plaques d'immatriculation tunisienne. Nous nous trouvons confronté à des plaques d'immatriculation « qui ne respectent » pas le modèle standard imposé par le service des mines. Le processus de reconnaissance est donc très difficile.

Les plaques d'immatriculation tunisiennes avec des variations de forme sont présentées dans la figure 6.



Figure 6. Plaques d'immatriculation tunisiennes

VII. Solution proposée

Nous allons concevoir un système de détection des véhicules et de reconnaissance des plaques d'immatriculation. Notre système doit être capable de reconnaître les plaques d'immatriculation de notre pays. Pour y parvenir, nous avons proposé un système basé sur l'approche de la reconnaissance de formes. De plus notre système doit fonctionner en temps réel, il ne doit pas être affecté par les perturbations qui peuvent se produire dans les images et les dommages mécaniques des plaques qui peuvent survenir.

Le processus de fonctionnement de notre système peut se composer des étapes suivantes :

La première étape concerne l'apprentissage tandis que la deuxième étape consiste à reconnaître les différentes formes en se basant sur le classifieur déjà construit dans la première phase.

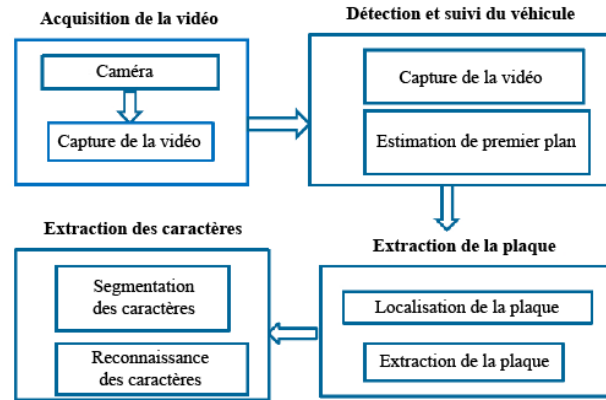


Figure 7.Fonctionnement du système

La figure 7 explique le fonctionnement détaillé du système qui agit ainsi :

- 1-Prendre une photo ou une vidéo.
- 2-Localiser la présence d'une plaque d'immatriculation dans l'image capturée ou dans la séquence vidéo.
- 3-Décoder la plaque d'immatriculation en utilisant la reconnaissance optique de caractère.

VIII. Planning prévisionnel

La clé principale de la réussite d'un projet est un bon planning. En effet, le planning aide à bien subdiviser le travail et séparer les tâches à réaliser, il offre une meilleure estimation et gestion de temps nécessaire pour chaque tâche. De plus, il donne assez de visibilité permettant d'estimer approximativement la date d'achèvement de chaque tâche.

Dans notre projet, nous avons estimé de réaliser notre application dans une durée approximative de trois mois. Le tableau ci-dessous montre le planning que nous avons adapté pour mener à bien notre réalisation des différentes parties du projet.

Tableau 1.Chronogramme des étapes de la réalisation

Mois	Février				Mars				Avril				Mai			
Semaines	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4
Etude préalable																
Réalisation																
Tests																
Rédaction																

Comme le montre le tableau ci-dessus, quatre principales phases peuvent être dégagés :

L'étude préalable: Le résultat de cette phase est la détermination des objectifs à atteindre dans notre future application en partant de l'existant.

Réalisation: Il s'agit de réaliser l'implémentation des programmes et effectuer les tests unitaires.

Test: Il s'agit de tester notre application.

Rédaction: Description détaillée de notre travail.

IX. Conclusion

Dans ce premier chapitre, nous avons défini le champ de notre étude afin de préciser nos objectifs. Nous avons expliqué le besoin de ce système de reconnaissance des plaques d'immatriculation, aussi nous avons fait abstraction de ce système, expliquer ses avantages vis-à-vis les facteurs qui peuvent influencer sur ses performances. Dans le chapitre qui suit nous présenterons la phase de la reconnaissance optique de caractères avec une étude de l'algorithme existant.

Chapitre 2 :

**Reconnaissance et
extraction de la plaque
d'immatriculation**

I. Introduction

Dans ce chapitre, nous allons présenter une API (Application Programming Interface), ainsi que les outils utilisés pour l'adapter à notre besoin. D'abord, nous parlons de la phase d'apprentissage du système de classification, ensuite nous expliquons le fonctionnement de l'API et enfin nous montrons le fonctionnement du système.

II. Apprentissage supervisé basé sur KNN

Dans ce travail, nous avons recours à la traduction d'images de plaque d'immatriculation en texte, pour y parvenir nous procédons tout d'abord à une étape d'apprentissage supervisé.

L'apprentissage supervisé est une technique d'apprentissage automatique où l'on cherche à produire automatiquement des règles à partir d'une base de données d'apprentissage contenant des « exemples » qui sont en général des cas déjà traités et validés.

En intelligence artificielle, l'algorithme KNN [1] (La méthode des k plus proches voisins) est une méthode d'apprentissage supervisé. Notre application tourne autour de cet algorithme que nous allons utiliser pour construire une application ROC (reconnaissance optique de caractères).

Dans ce cadre, nous disposons d'une base de données d'apprentissage constituée de N couples « entrée-sortie ». Pour estimer la sortie associée à une nouvelle entrée x , la méthode des k plus proches voisins consiste à prendre en compte (de façon identique) les k échantillons d'apprentissage dont l'entrée est la plus proche de la nouvelle entrée x , selon une distance à définir.

La figure 8 représente un exemple d'un arbre de décision d'un problème de classification.

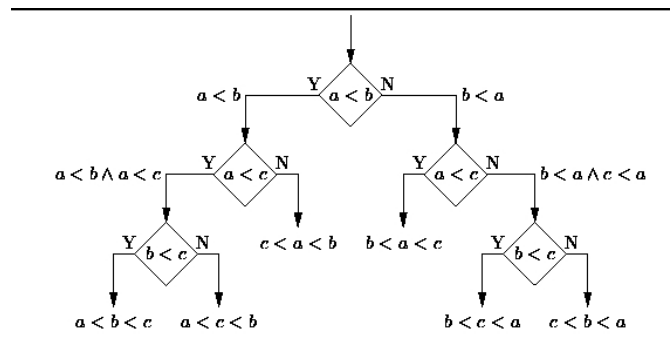


Figure 8. Arbres de décision de l'algorithme KNN

III. La bibliothèque OpenCV

OpenCV [2] (Open Computer Vision) est une bibliothèque graphique libre, initialement développée par Intel, spécialisée dans le traitement d'images en temps réel.

La société de robotique Willow Garage, la société Isteez et Intel se sont succédé au support de cette bibliothèque. Elle est distribuée sous licence BSD.

OpenCV a été utilisée dans de nombreuses applications. Cette bibliothèque propose la plupart des opérations classiques en traitement des images. Elle propose un nombre important d'outils. Elle met également à disposition quelques fonctions d'interfaces graphiques, comme les curseurs à glissière, les contrôles associés aux événements souris. Par ailleurs elle permet d'offrir des opérations de traitement de vidéos.

Au cours de notre recherche nous avons trouvés une API qui peut nous aider à atteindre notre objectif mais il faut l'adapter à notre besoin. Cette API est disponible dans plusieurs langages tels que C++, C#, Python, Visual basic.

Nous nous sommes intéressés à la version Python puisqu'elle s'avère être la plus optimale en temps d'exécution comme le montre le tableau 2.

Tableau 2. Tableau comparatif en temps d'exécution

Langage	Grayscale (ms)	Binarisation (ms)
Python	9	6
C++	10	8
Visual Basic	33	17

En effet nous cherchons l'optimum et le traitement en quasi-temps réel, notre choix s'est orienté vers l'utilisation et l'adaptation de cette API écrite en Python couplé avec le langage Java pour l'utilisation. Le choix du langage Java est pris par la société grâce à ses nombreux avantages.

IV. Analyse de l'API existante

Dans cette section, nous allons analyser l'API existante, la critiquer et proposer des modifications.

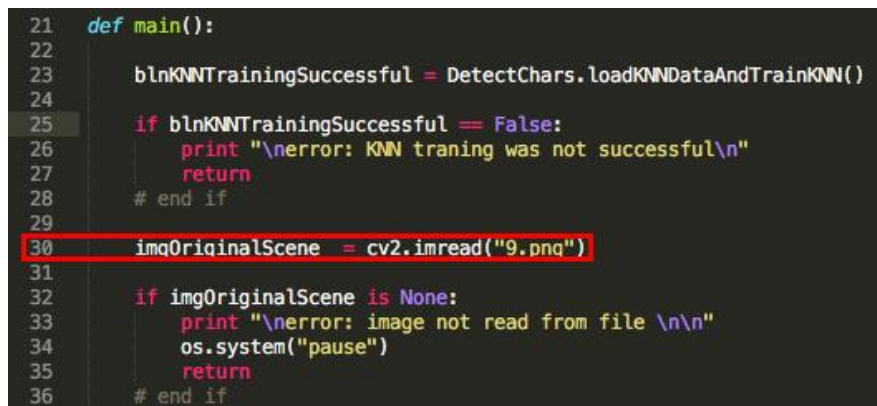
IV.1 Etude de l'API existante

Dans cette section, nous présentons l'API existante, son utilisation et son fonctionnement.

IV.1.1 Présentation

GitHub [3] est un service web d'hébergement et de gestion de développement de logiciels, dont on a téléchargé l'API qu'on va la modifier selon notre besoin.

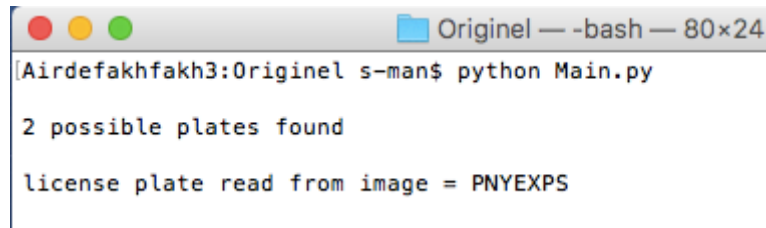
En effet comme le montre la figure 9, l'API ne prend aucun paramètre en entrée, l'image du véhicule source sera précisée dans le code avant l'exécution.



```
21 def main():
22
23     blnKNNTrainingSuccessful = DetectChars.loadKNNDataAndTrainKNN()
24
25     if blnKNNTrainingSuccessful == False:
26         print "\nerror: KNN training was not successful\n"
27         return
28     # end if
29
30     imgOriginalScene = cv2.imread("9.png")
31
32     if imgOriginalScene is None:
33         print "\nerror: image not read from file \n\n"
34         os.system("pause")
35         return
36     # end if
```

Figure 9.L'entrée du programme

La figure 11 montre les entrées et les sorties du programme qui est considéré comme une boîte noire. L'utilisation de l'API se fait à partir du Terminal (Voir Figure 10).

A terminal window titled 'Originel — -bash — 80x24' with standard macOS window controls (red, yellow, green buttons). The prompt is '[Airdefakhfakh3:Originel s-man\$' and the command 'python Main.py' has been executed. The output shows '2 possible plates found' followed by 'license plate read from image = PNYEXPS' on the next line.

```
[Airdefakhfakh3:Originel s-man$ python Main.py  
2 possible plates found  
license plate read from image = PNYEXPS
```

Figure 10.Utilisation de Main.py

IV.1.2 Fonctionnement

La figure 12 donne le fonctionnement de l'API, que nous allons détailler dans la section VI, de l'image vers le texte.

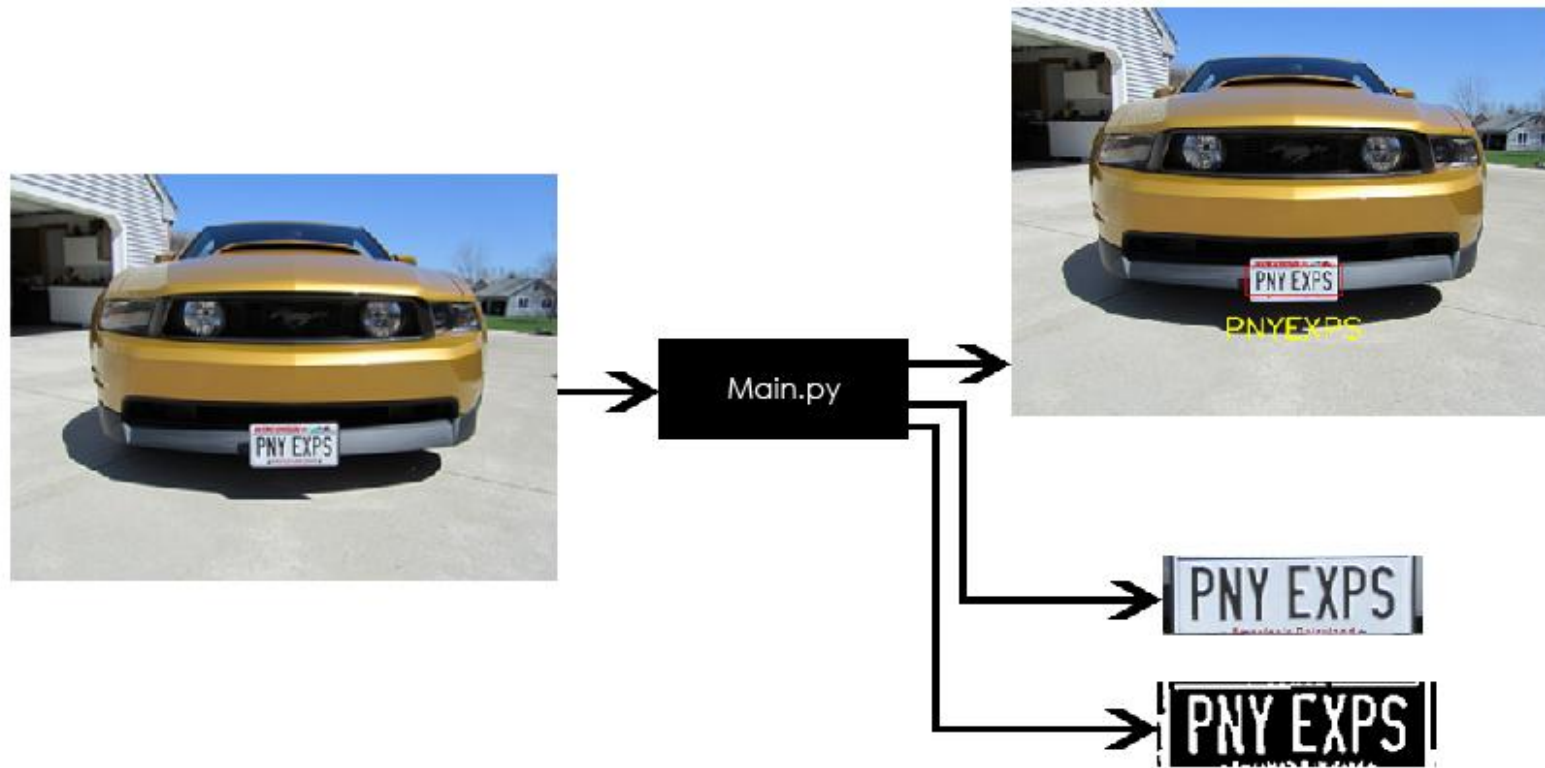


Figure 11. Les E/S de l'API d'origine avec un véhicule étranger

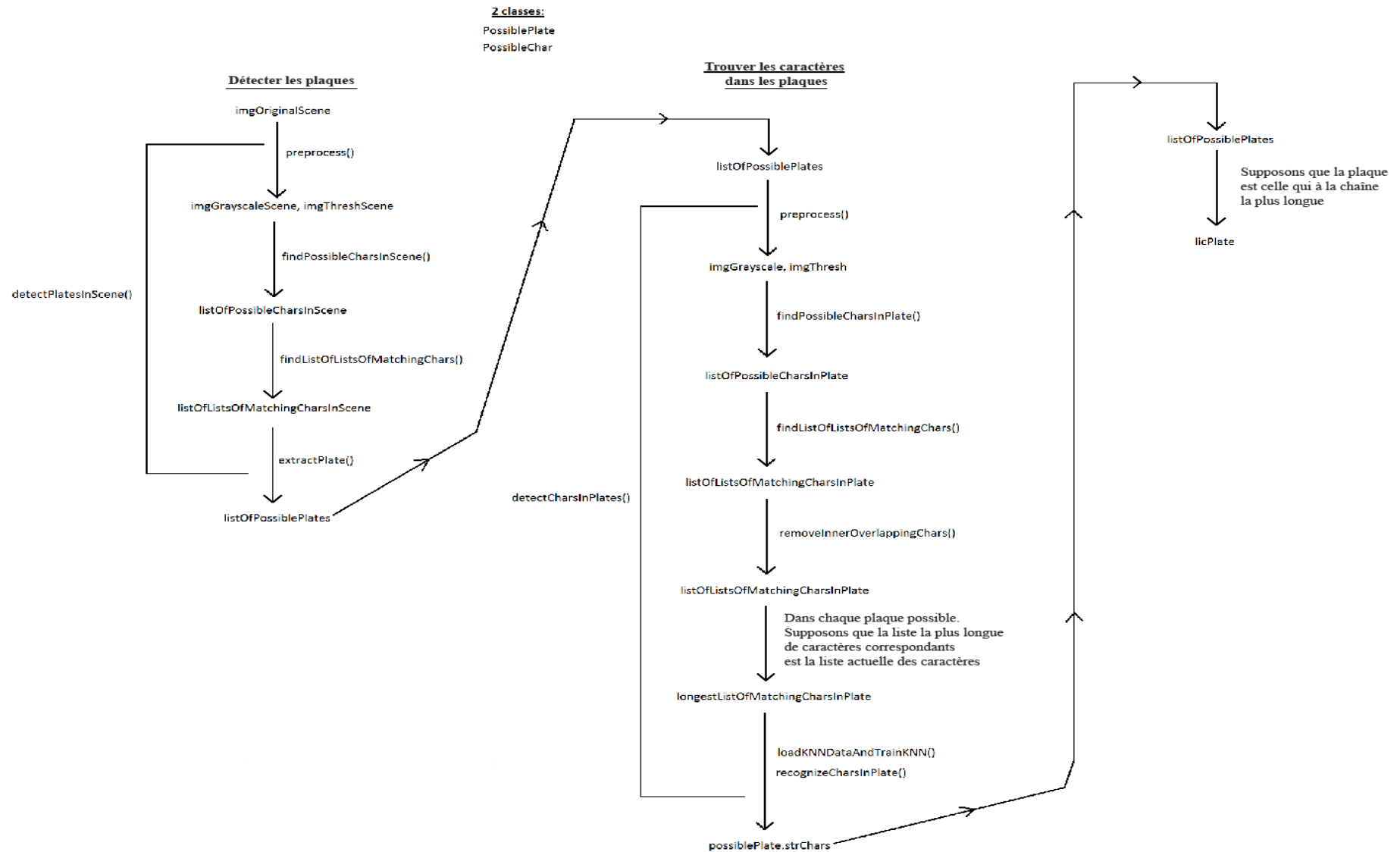


Figure 12. Fonctionnement de l'API

IV.2 Etude critique

Comme nous pouvons le constater, l'API actuelle que nous avons citée précédemment présente certaines limites qui sont :

- Incompatibilité avec les plaques d'immatriculation tunisiennes.
- Des résultats en sortie qui sont erronés (Voir Figure 16).
- L'image en entrée doit être passé en paramètre.
- Utilisation complexe.

V. Contribution

Dans cette section, nous présentons notre contribution qui se résume à la modification apportée à l'API.

V.1 Apprentissage du système de classification

Pour que le système détecte les plaques et les caractères de la norme tunisienne, nous avons construit une base de données d'apprentissage contenant plus que 1000 exemples. La figure 13 donne un exemple illustratif, ainsi nous avons généré grâce à OpenCV deux fichiers sur lesquels l'algorithme KNN se base (Voir Figure 14).



Figure 13.Extrait du dictionnaire d'apprentissage

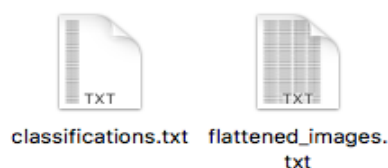


Figure 14.Fichiers de classification

V.2 Utilisation de l'API

Nous avons modifié l'API de façon à prendre le chemin relatif de l'image. (Voir Figure 15)

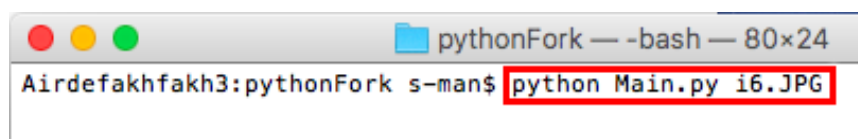


Figure 15.Utilisation de l'API résultante

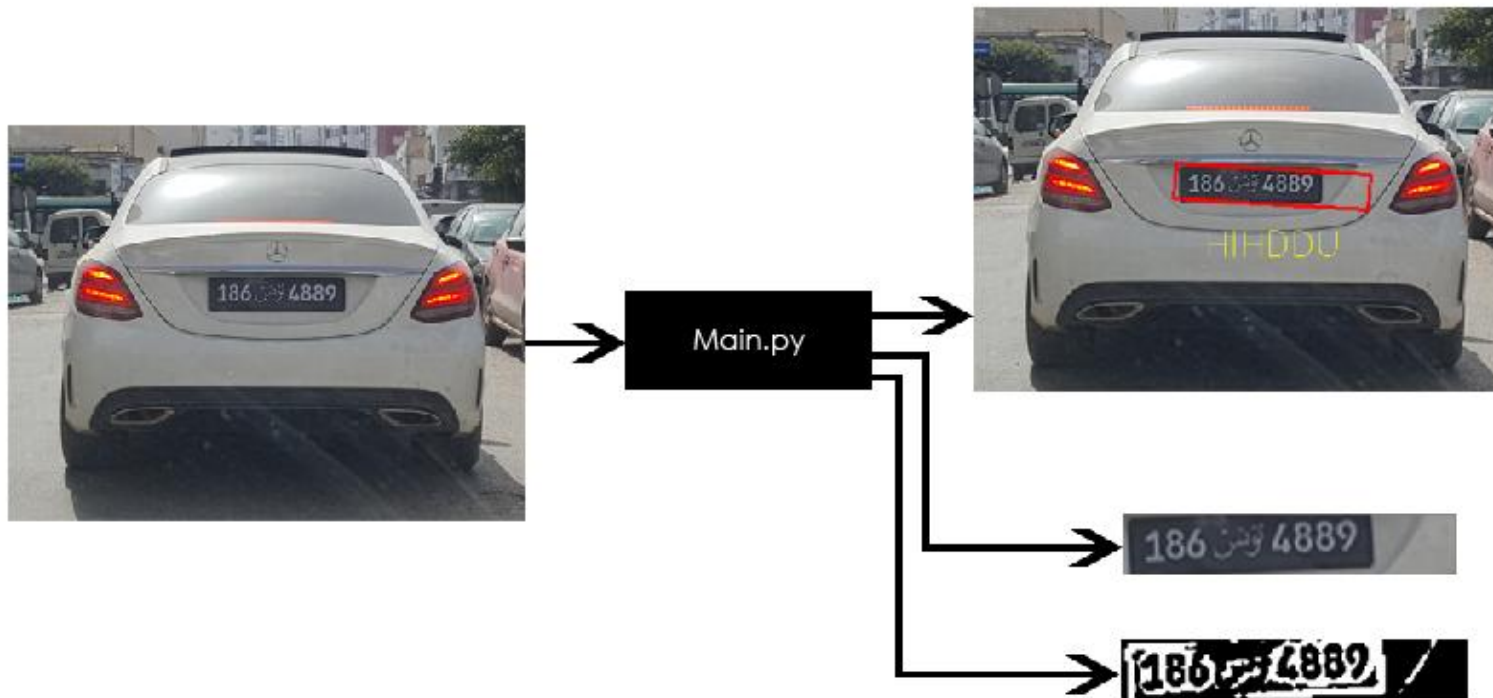


Figure 16.Exécution de l'API d'origine avec une plaque d'immatriculation tunisienne

VI. Processus de fonctionnement

Le schéma ci-dessous détaille les étapes de l'extraction de la plaque d'immatriculation.

imgOriginalScene



preprocess()

imgGrayscaleScene, imgThreshScene



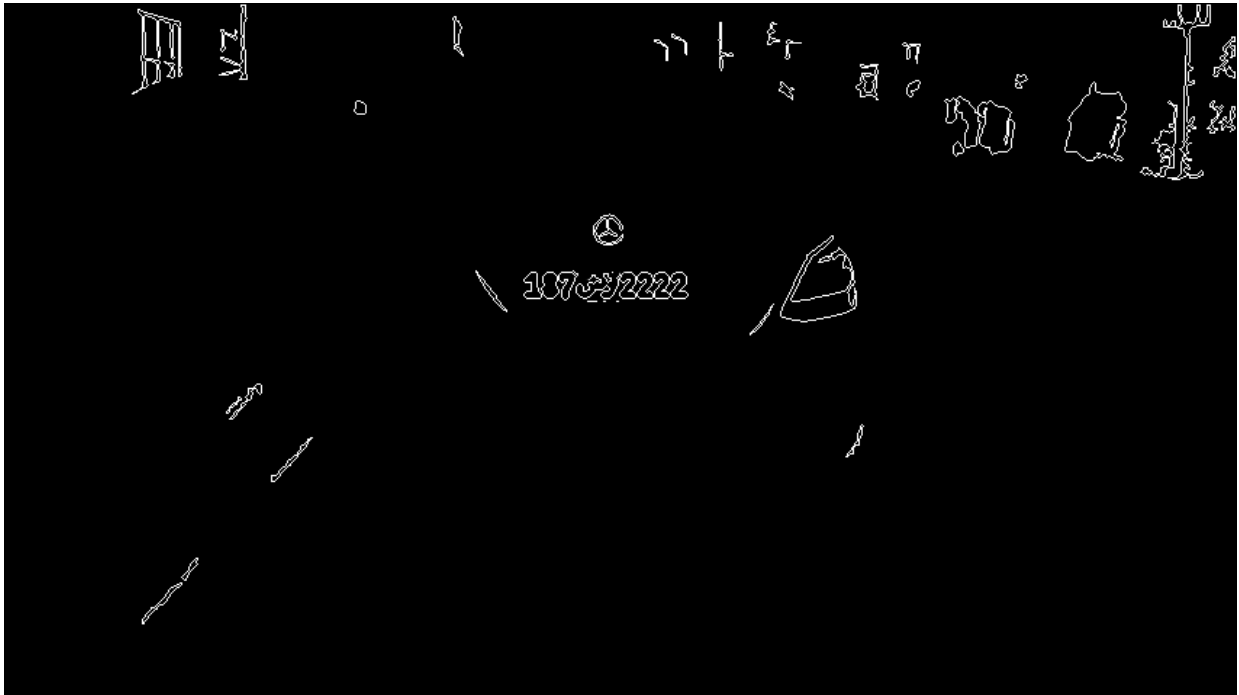


`findPossibleCharsInScene()`

all contours

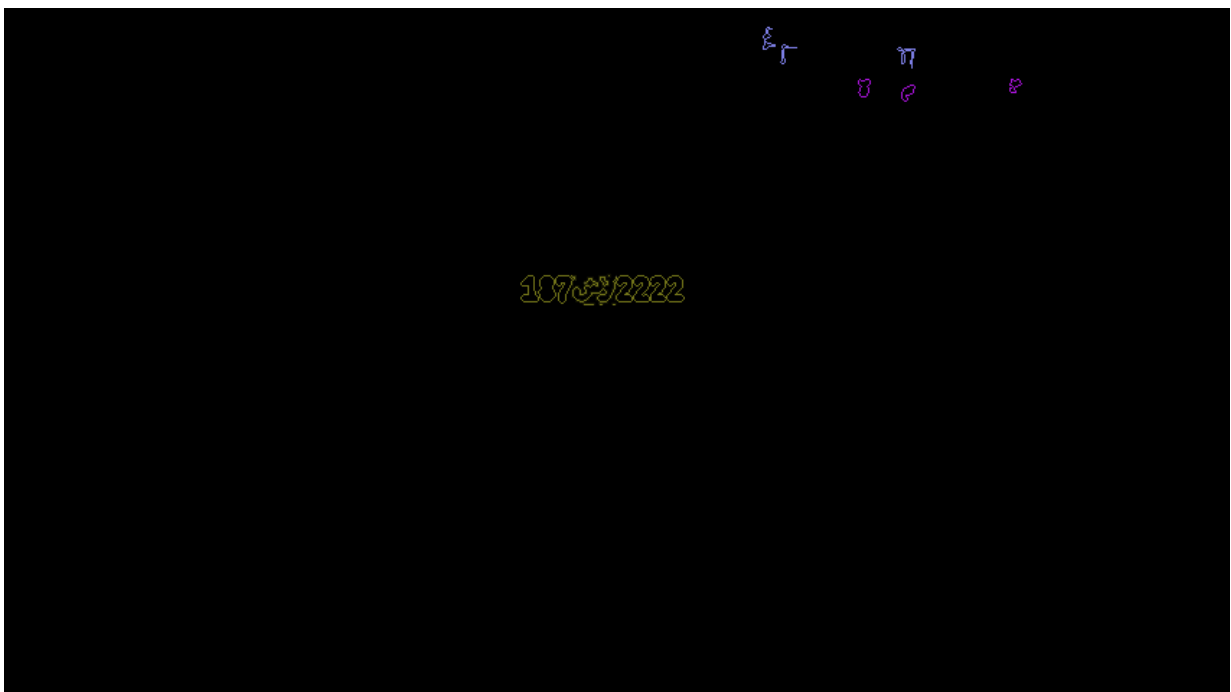


vectorOfPossibleCharsInScene



findVectorOfVectorsOfMatchingChars()

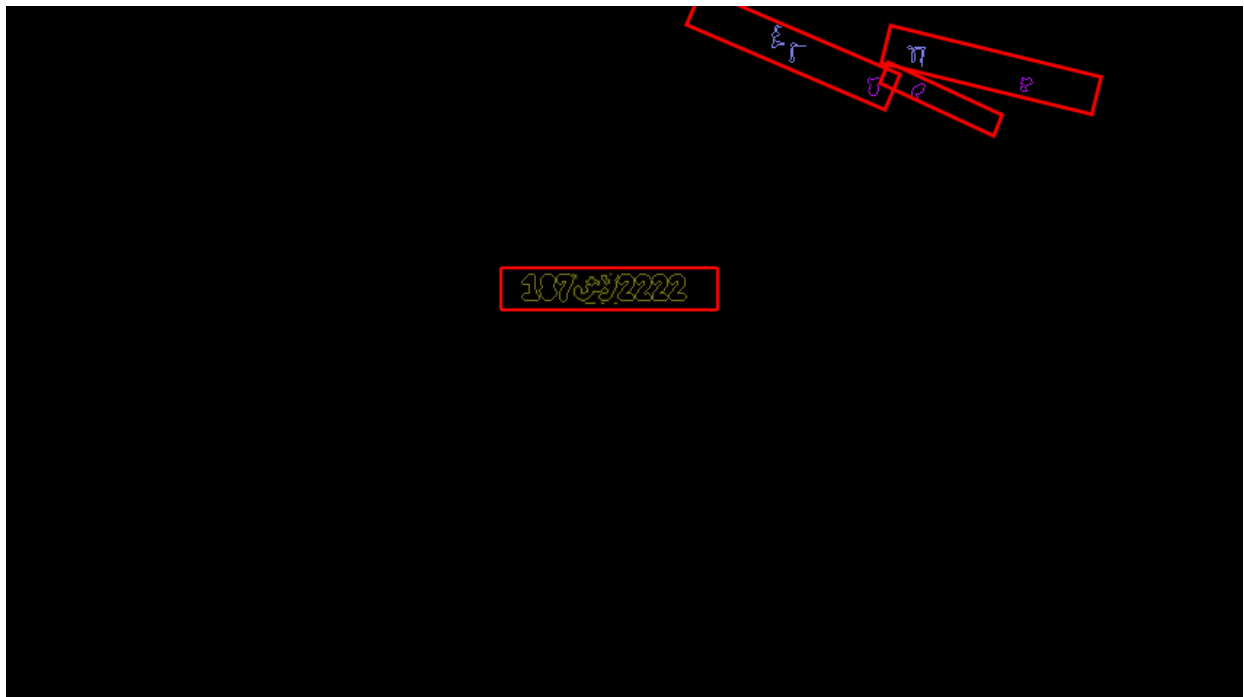
vectorOfVectorsOfMatchingCharsInScene





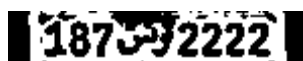
extractPlate()

vectorOfPossiblePlates



preprocess()

imgGrayscale, imgThresh



findPossibleCharsInPlate()

vectorOfPossibleCharsInPlate



findVectorOfVectorsOfMatchingChars()

vectorOfVectorsOfMatchingCharsInPlate



removeInnerOverlappingChars()

vectorOfVectorsOfMatchingCharsInPlate



Dans chaque plaque possible.
Supposons que la liste la plus longue
de caractères correspondants
est la liste actuelle des caractères

longestVectorOfMatchingCharsInPlate



recognizeCharsInPlate()



La chaîne de caractères trouvée dans la plaque numéro 0 est "187TN2222"

La chaîne de caractères trouvée dans la plaque numéro 1 est "L1"

possiblePlate.strChars



Supposons que la plaque
est celle qui à la chaîne
la plus longue

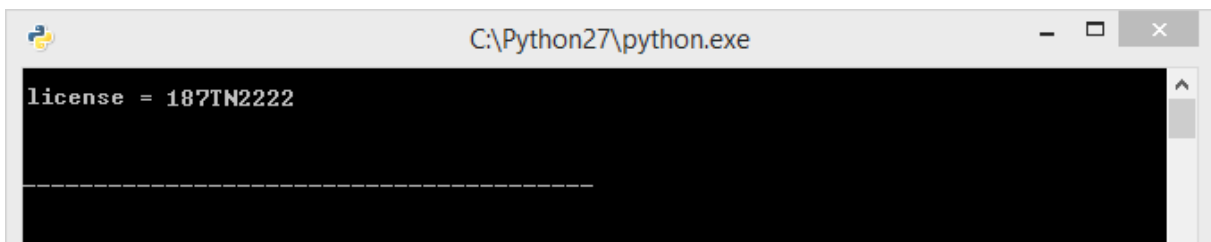


Figure 17. Résultat fournie en sortie contenant le texte de la plaque d'immatriculation

VII. Les Wrappers

Notre application est écrite en Java. Afin d'utiliser les bytes codes Python, nous avons recouru à l'implémentation d'un Wrapper. Un Wrapper est un programme dont la fonction est capable d'appeler une autre fonction, en faisant abstraction sur le langage de cette dernière et de stocker ses résultats.

En programmation orientée objet, cette notion est aussi connue sous le nom de méthode de délégation.

Le Wrapper dont le code est montré par la figure 18 utilise les E/S standards du système pour fonctionner.

```

public static PythonResult extract(String filename) {
    Process pyEx = null;
    long debut = System.currentTimeMillis();
    try {
        pyEx = Runtime.getRuntime().exec("python /tryWithJava/Main.pyc " + filename);
    } catch (IOException e) {
        e.printStackTrace();
    }
    long fin = System.currentTimeMillis();
    try {
        pyEx.waitFor();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    Scanner scanner = new Scanner(pyEx.getInputStream());
    String resultat = scanner.nextLine();
    if (resultat.contains("license")) {
        PythonResult pr = new PythonResult();
        File image = new File("tryWithJava/outputs/imgOriginalScene.png");
        int rnd = (int) (Math.random() * 10000);
        String newFile = image.getAbsolutePath().replace("imgOriginalScene", "resultat" + rnd);
        image.renameTo(new File(newFile));
        pr.setImgSuccesful(newFile);
        image = new File("tryWithJava/outputs/imgPlate.png");
        rnd = (int) (Math.random() * 10000);
        newFile = image.getAbsolutePath().replace("imgPlate", "resultat" + rnd);
        image.renameTo(new File(newFile));
        pr.setImgPlate(newFile);
        pr.setNumPlate(resultat.substring(resultat.lastIndexOf("=") + 1, resultat.length()));
        pr.setNumPlate(Main.traiter(pr.getNumPlate()));
        pr.setExectime(fin - debut);

        return pr;
    }
    return null;
}

```

Figure 18. Java Wrapper implémenté

Pour enregistrer les résultats, nous avons implémenté la classe "PythonResult" montrée par la figure 19 qui possède les attributs suivants :

- imgPlate : Contient le chemin absolu de la plaque recadrée.
- imgSuccesful : Contient l'image du véhicule avec son numéro de plaque écrit dessus.
- numPlate : Contient le texte extrait de la plaque.
- execTime : Contient le temps d'exécution de la délégation et l'extraction.

<<Java Class>>	
PythonResult	
-imgPlate:	String
-imgSuccesful:	String
-numPlate:	String
-execTime:	long

Figure 19. La classe PythonResult

VII.1 Exemple illustratif

Nous supposons dans notre exemple que l'image se situe au chemin `"/home/image.png"`. Le tableau 3 explique brièvement le fonctionnement du système. Après exécution nous obtiendrons :

La sortie sur la console du byte code est montrée par la figure 20.

```
Airdefakhfakh3:pythonFork s-man$ python Main.py b1.png  
license = 1812212
```

Figure 20.Utilisation de l'API modifiée

L'image disponible au chemin `"/home/image.png"` est visible dans la figure 21



Figure 21.Image d'entrée

L'image disponible au chemin `"/tryWithJava/outputs/resultat1404.png"` est visible dans la figure 22.



Figure 22. Image du véhicule fournie en sortie

L'image disponible au chemin `"/tryWithJava/outputs/resultat4234.png"` est visible dans la figure 23.



Figure 23. Image de la plaque fournie en sortie

Tableau 3.Tournage du Wrapper

Méthodes	début	fin	résultat	pythonResult.i mgPlate	pythonResult.imgSu ccesful	pythonResult.num PLate	pythonResult .execTime
long debut = System.currentTimeMillis();	14949670 21303	-	-	-	-	-	-
long fin = System.currentTimeMillis();	14949670 21303	14949670 21311	-	-	-	-	-
String resultat = scanner.nextLine();	14949670 21303	14949670 21311	license = 187TN2222	-	-	-	-
pr.setImgSuccesful(newFile);	14949670 21303	14949670 21311	license = 187TN2222	-	/tryWithJava/outputs /resultat1404.png	-	-
pr.setImgPlate(newFile);	14949670 21303	14949670 21311	license = 187TN2222	/tryWithJava/out puts/resultat434 .png	/tryWithJava/outputs /resultat1404.png	-	-
pr.setNumPlate(resultat.substring(result at.lastIndexOf("=") + 1, resultat.length()));	14949670 21303	14949670 21311	license = 187TN2222	/tryWithJava/out puts/resultat434 .png	/tryWithJava/outputs /resultat1404.png	187TN2222	-
pr.setExectime(fin - debut);	14949670 21303	14949670 21311	license = 187TN2222	/tryWithJava/out puts/resultat423 4.png	/tryWithJava/outputs /resultat1404.png	187TN2222	8

VIII. Conclusion

Dans ce deuxième chapitre, nous avons analysé l'API existante, montré ses limites et proposé une solution pour son exploitation.

Dans le chapitre suivant, nous présenterons la phase de la détection des véhicules en mouvement à partir d'une vidéo.

Chapitre 3 :

Traitement vidéo

I. Introduction

Une vidéo est une succession d'images, pour traiter une vidéo, il suffit tout simplement de faire des traitements sur ses images, on a comme objectif la détection et l'extraction du véhicule afin de pouvoir par la suite reconnaître sa plaque d'immatriculation.

II. Les trois approches de détection d'un véhicule

Il existe trois approches pour la détection d'un véhicule :

- Position du véhicule connu à l'avance.
- Détection et extraire tous les véhicules probables.
- Détection des éventuelles plaques d'immatriculation.

II.1 Position du véhicule connu à l'avance

Cette approche consiste à fixer la position du véhicule (Voir Figure 24). Dès qu'un véhicule passe (Voir Figure 25).









Figure 25.Position prévue pour le passage du véhicule



Figure 24.Un véhicule qui passe comme prévu

Nous faisons l'extraction de l'image contenue dans le cadre rouge de la figure 24 respectivement celle qui coïncide avec les mêmes coordonnées dans la figure 25. Le tableau 4 prouve qu'il y a une différence apparente de plus que 80%, donc c'est un véhicule.

Tableau 4. Tableau comparatif

Image extraite	Image extraite
	
Flou gaussien avec un taux de 90 %	Flou gaussien avec un taux de 90 %
	
Binarisation	Binarisation
	

II.2 Détecter et extraire tous les véhicules probables

Ces deux algorithmes sont principalement utilisés pour le dénombrement des véhicules comme le montre la figure 26.

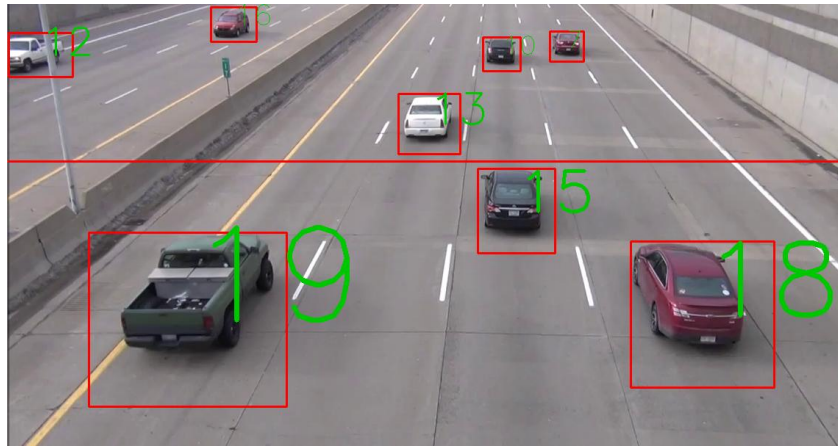


Figure 26. Dénombrement des véhicules

II.2.1 Algorithme de Haarcascade

La méthode de Haarcascade [4] consiste à trouver des régions rectangulaires dans l'image donnée qui sont susceptibles de contenir des objets pour lesquels la cascade a été formée et renvoie ces régions sous la forme d'une séquence de rectangles. La fonction scanne l'image plusieurs fois à différentes échelles. Il peut également appliquer quelques heuristiques pour réduire le nombre de régions analysées (Voir Figure 27). Après avoir procédé et recueilli les rectangles candidats, il les regroupe et retourne une séquence de rectangles moyens pour chaque groupe suffisamment grand.

```

trainvideo — opencv_traincascade -data haar -vec samples.ve...

NEG current samples: 332
NEG current samples: 65933
NEG current samples: 823
NEG current samples: 987
NEG current samples: 1145
NEG current samples: 1302

NEG count : acceptanceRatio    1436 : 1.44088e-05
Precalculation time: 168
+-----+
|  N  |  HR  |  FA  |
+-----+
|  1  |    1 |    1 |
+-----+
|  2  |    1 |    1 |
+-----+
|  3  |    1 |    1 |
+-----+
|  4  |    1 |    1 |
+-----+
|  5  |    1 |    1 |
+-----+
|  6  |    1 |    1 |
+-----+
|  7  |    1 | 0.86351 |
+-----+
|  8  |    1 | 0.880919 |
+-----+
|  9  |    1 | 0.608635 |
+-----+
| 10  |    1 | 0.575905 |
+-----+
| 11  |    1 | 0.339833 |
+-----+
END>
Training until now has taken 0 days 11 hours 9 minutes 15 seconds.

===== TRAINING 15-stage =====
<BEGIN
POS count : consumed    574 : 574

```

Figure 27. Apprentissage de la base requise par Haarcascade

II.2.2 Algorithme de HOG Descriptor

Un histogramme de gradient orienté (HOG) est une caractéristique utilisée en vision par ordinateur pour la détection d'objet. Cette technique calcule des histogrammes locaux de l'orientation du gradient sur une grille dense, c'est-à-dire sur des zones régulièrement réparties sur l'image.

II.2.3 Différences entre Haarcascade et HOG Descriptor

La méthode de l'histogramme de gradient orienté (HOG) s'est montrée particulièrement efficace pour la détection de personnes contrairement à la méthode de Haarcascade qui s'est avérée meilleure pour la détection d'objets proposés et plus rapide.

II.3 Détection des éventuelles plaques d'immatriculation

Les deux approches présentées dans les sections II.1 et II.2 n'ont pas satisfait nos besoins, nous avons développé une approche ultime qui prend en considération les éléments suivants :

-La minimisation du temps d'exécution.

-Optimisation de l'espace mémoire.

-Précision de la détection.

Cette approche consiste à ne détecter que les plaques en détectant tous les rectangles possibles et en appliquant plusieurs critères pour éliminer les rectangles qui ne contiennent pas de plaque d'immatriculation (Voir Figure 28). Les critères de sélection sont les suivantes :

-Le rapport largeur / hauteur doit être compris entre 3 et 5.

-La hauteur ne doit pas dépasser le 1/5 de la hauteur de l'image.

-La hauteur ne doit pas être plus petite que le 1/50 de l'image.

Pour détecter les rectangles possibles dans l'image, il faut passer par ces étapes :

-Niveau de gris

-Flou gaussien

-Binarisation

-Repérer les contours



Figure 28. Les matricules encadrées

III. Détection de la plaque d'immatriculation

Dans cette section, nous expliquons le fonctionnement de l'approche utilisée et nous montrons un exemple illustratif.

III.1 Fonctionnement

Dans cette section nous nous intéressons à la détection de la plaque d'immatriculation en mouvement à partir d'une vidéo en utilisant l'approche citée dans la section II.3. Cette approche dont le code est divulgué dans la figure 29 fonctionne comme suit :

Il faut tracer une ligne qui servira de capteur dans l'image extraite de la vidéo, si un rectangle est en mouvement et qu'il traverse la ligne, c'est probablement une plaque d'immatriculation. Dans le cas il s'avère être un parasite, la délégation au Wrapper ne détectera aucune plaque ni texte et l'ignorera.

```
while (cap.read(image)) {
    aux = image.clone();
    label.setIcon(Main.matToLabel(image.clone()));
    Mat imageHSV = new Mat(image.size(), CvType.CV_8UC4);
    Mat imageBlurr = new Mat(image.size(), CvType.CV_8UC4);
    Mat imageA = new Mat(image.size(), CvType.CV_32F);
    Imgproc.cvtColor(image, imageHSV, Imgproc.COLOR_BGR2GRAY);
    Imgproc.GaussianBlur(imageHSV, imageBlurr, new Size(5, 5), 2);
    Imgproc.adaptiveThreshold(imageBlurr, imageA, 255, Imgproc.ADAPTIVE_THRESH_MEAN_C, Imgproc.THRESH_BINARY, 7, 5);
    List<MatOfPoint> contours = new ArrayList<MatOfPoint>();
    Imgproc.findContours(imageA, contours, new Mat(), Imgproc.RETR_LIST, Imgproc.CHAIN_APPROX_SIMPLE);
    for (int i = 0; i < contours.size(); i++) {
        if (Imgproc.contourArea(contours.get(i)) > 50) {
            Rect rect = Imgproc.boundingRect(contours.get(i));
            if ((float) (rect.width / rect.height) >= 3 && (float) (rect.width / rect.height) <= 5
                && rect.height > image.rows() / 50 && rect.height < image.rows() / 4
            ) {
                Imgproc.rectangle(image, new Point(rect.x, rect.y),
                    new Point(rect.x + rect.width, rect.y + rect.height), new Scalar(0, 0, 255));
                double m = (rect.y + rect.y + rect.height) / 2;
                if (rect.contains(new Point(rect.x, image.rows() / 1.5))) {
                    System.out.println("voiture");
                    String res = "/Users/s-man/Desktop/tryWithJava/inputs/video"
                        + (int) (Math.random() * 10000) + ".jpg";
                    Mat a = new Mat();
                    Imgproc.resize(aux.submat(rect), a, new Size(500, 109));
                    Imgcodecs.imwrite(res, a);
                }
            }
        }
    }
}
```

Figure 29. Code source de l'approche de détection de plaques utilisée

III.2 Exemple illustratif

En premier temps, nous présentons un exemple de détection de rectangles, nous nous abstiendrons de le reproduire ultérieurement. Voici le schéma explicatif:

Image



```
Imgproc.cvtColor(image, imageHSV, Imgproc.COLOR_BGR2GRAY);
```



ImageHSV



`Imgproc.GaussianBlur(imageHSV, imageBlurr, new Size(5, 5), 2);`



ImageBlurr



```
Imgproc.adaptiveThreshold (imageBlurr, imageA, 255,Imgproc.ADAPTIVE_THRESH_MEAN_C,Imgproc.THRESH_BINARY, 7, 5);
```



imageA



```
Imgproc.findContours (image, contours, new Mat(), Imgproc.RETR_LIST, Imgproc.CHAIN_APPROX_SIMPLE);
```



Pour chaque contour nous appliquons :

```
Imgproc.rectangle(image, new Point(rect.x, rect.y), new Point(rect.x + rect.width, rect.y + rect.height), new Scalar(0, 0, 255));
```

sur l'image originelle.

Image

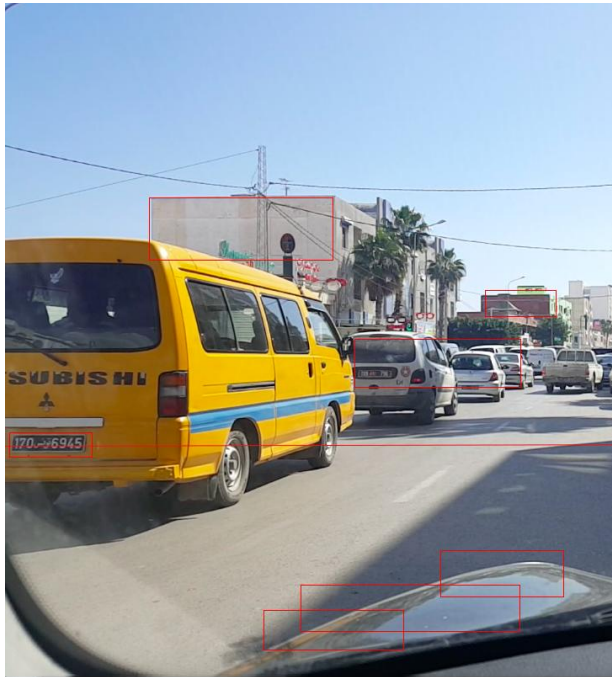


A présent nous nous concentrons dans l'illustration d'un exemple réel d'une vidéo qui détectera les plaques:

L'instant 00:00



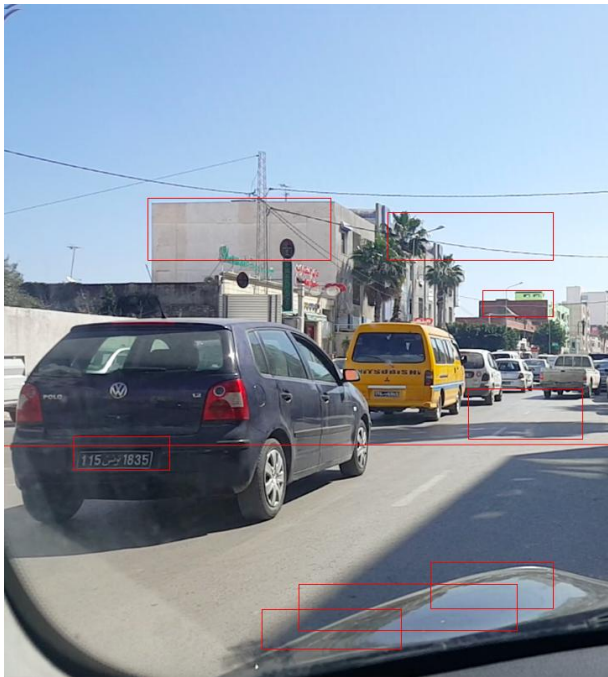
L'instant 00:13



L'instant 00:32



L'instant 01:24



Ainsi les plaques détectées seront traitées au fur et à mesure lors de la détection.

IV. Conclusion

Ce chapitre présente les différentes méthodes de détection des véhicules ainsi que leurs performances.

Dans le chapitre suivant, nous explorons l'architecture logicielle adoptée, le diagramme généré par rétro-ingénierie, ainsi que des captures de notre application réalisé dans tous ses modes de fonctionnement.

Chapitre 4 :
Rétro-ingénierie et présentation
de l'application

I. Introduction

Ce chapitre comporte trois sections, la première justifie l'architecture sur laquelle nous avons développé notre application, la seconde s'occupe de la phase de la rétro-ingénierie et le diagramme de classes et la dernière donne une vue générale sur les interfaces graphiques de l'application réalisée.

II. Architecture logicielle

Une application Java possède souvent une architecture 3-tiers (Voir Figure 34) :

- ✓ la couche DAO (Data Access Object) s'occupe de l'accès aux données et leurs persistance au sein d'un fichier XML dans notre cas.
- ✓ La couche métier implémente les algorithmes "métier" de l'application. Cette couche est indépendante de toute forme d'interface avec l'utilisateur. Elle doit ainsi pouvoir être testée en-dehors de l'interface et notamment avec une interface console. C'est généralement la couche la plus stable de l'architecture. Elle ne change pas si on change l'interface utilisateur ou la façon d'accéder aux données nécessaires au fonctionnement de l'application.
- ✓ La couche interface utilisateur qui est l'interface graphique qui permet à l'utilisateur de piloter l'application et d'en recevoir des informations.

Les couches métier et DAO sont normalement utilisées via des interfaces Java. Ainsi la couche métier ne connaît de la couche DAO que son (ou ses) interface(s) et ne connaît pas les classes les implémentant. C'est ce qui assure l'indépendance des couches entre-elles : changer l'implémentation de la couche DAO n'a aucune incidence sur la couche métier tant qu'on ne touche pas à la définition de l'interface de la couche DAO. Il en est de même entre les couches interface utilisateur et métier.

III. Diagramme de classes par rétro-ingénierie

Le diagramme de classes est un schéma structurel pour présenter les classes et les interfaces des systèmes ainsi que les différentes relations entre celles-ci. Le diagramme de classes dans la figure 33 est généré par une extension de l'IDE Eclipse nommée ObjectAid[5].

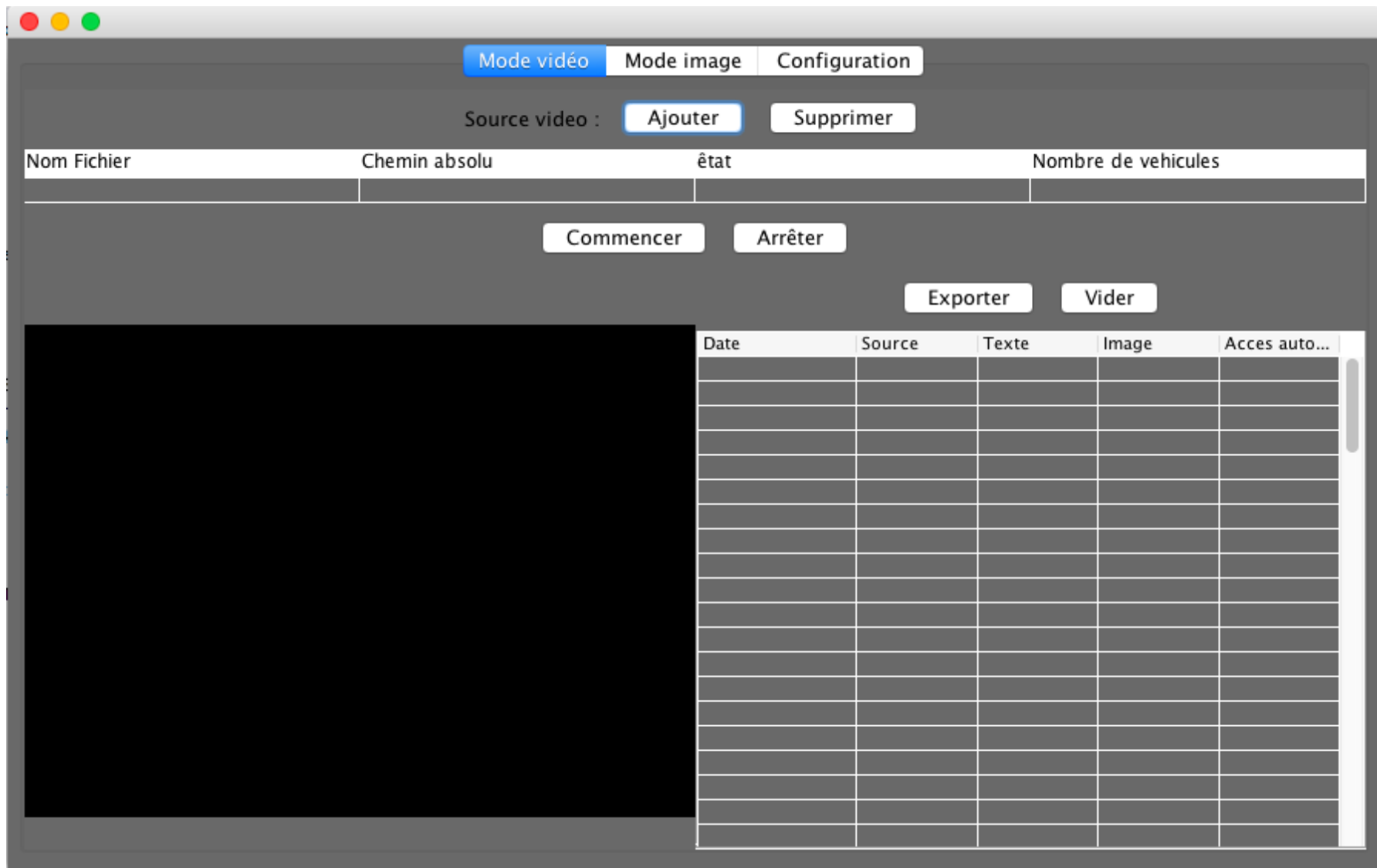


Figure 30. Onglet mode vidéo

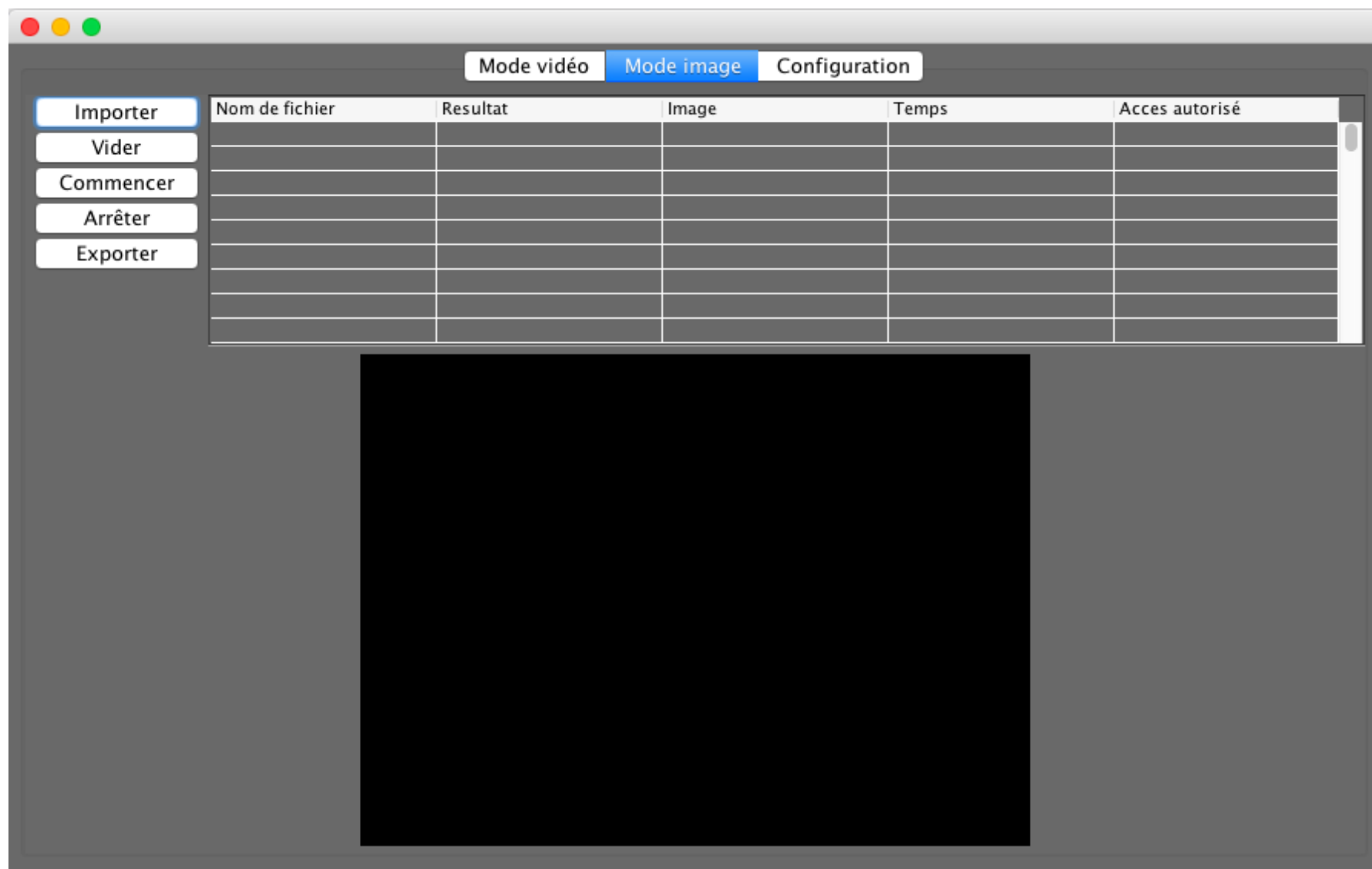


Figure 31. Onglet mode image

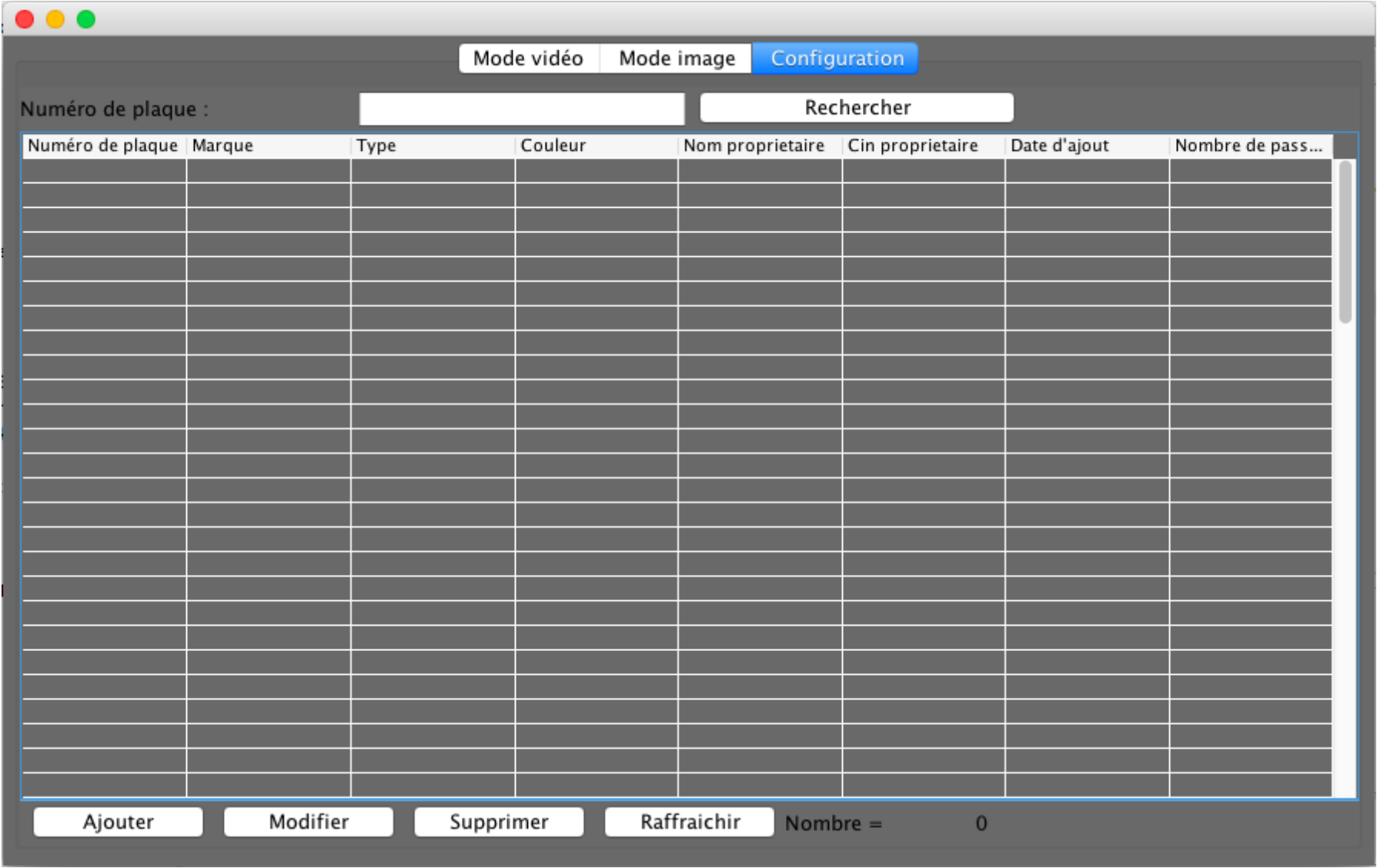


Figure 32.Onglet configuration

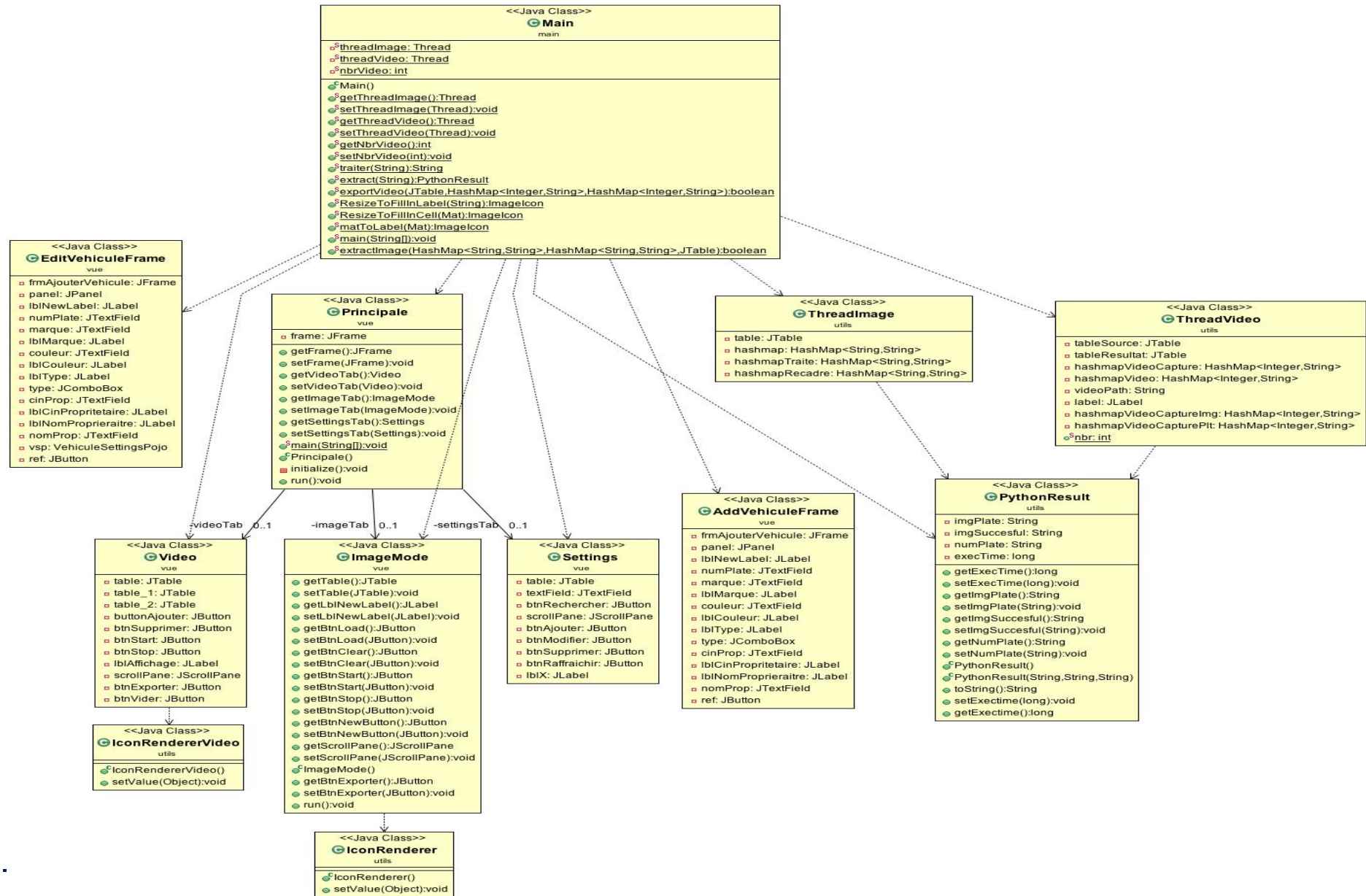


Figure 33. Diagramme de classes par rétro-ingénierie

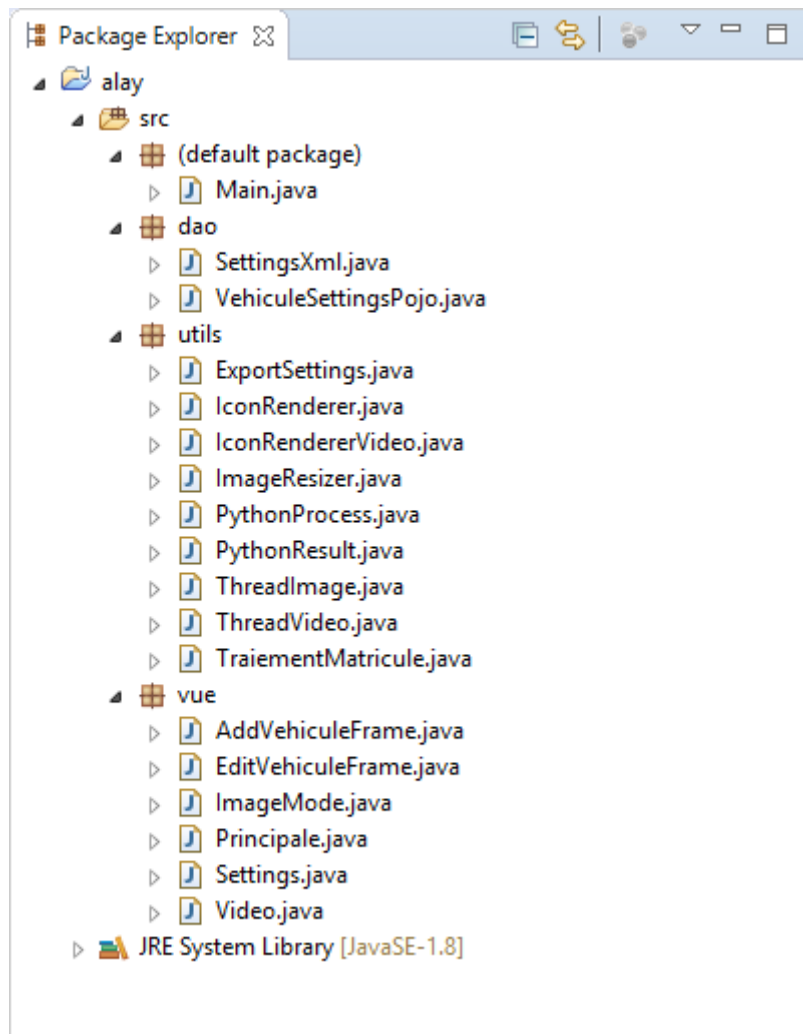


Figure 34. Hiérarchie du projet

IV. Présentation de l'application

Dans cette partie nous allons exposer les différentes interfaces graphiques qui permettent de nous mettre dans les conditions réelles d'utilisation de l'application.

Nous avons 3 onglets nommés respectivement Mode Vidéo, Mode Image et Configuration.

-Mode Vidéo : Dans cet onglet, l'utilisateur aura la possibilité d'ajouter ou supprimer la vidéo. Lancer ou arrêter le processus de traitement grâce aux deux boutons 'Start' et 'Stop'. Après avoir extrait les plaques d'immatriculation des véhicules contenus dans la vidéo, l'utilisateur peut exporter les résultats qui vont être stockés dans un fichier HTML sous format d'un tableau contenant l'image du véhicule, sa plaque recadrée et son texte chacun dans une colonne. (Voir Figure 30)

-Mode Image : Dans cet onglet, l'utilisateur peut choisir un répertoire contenant des images des véhicules que notre application va le balayer et faire tous les traitements de façon automatique. Avec chaque véhicule traité, nous obtiendrons une image de la plaque recadrée et son texte avec le temps de traitement exprimé en ms (Millisecondes). L'utilisateur peut aussi exporter les résultats de façon identique comme dans le Mode Vidéo. (Voir Figure 31)

-Configuration : Dans cet onglet, l'utilisateur peut chercher un véhicule d'après son numéro de plaque d'immatriculation, les informations relatives (numéro de la plaque, marque, nombre de passage, ...) vont être affichés dans un tableau. L'utilisateur peut aussi ajouter, modifier ou supprimer un véhicule de façon manuelle. (Voir Figure 32)

V. Conclusion

Dans ce chapitre, nous avons expliqué notre choix de l'architecture 3 tiers, dévoilé le diagramme de classes généré par rétro-ingénierie et montré un aperçu de l'application finale.

Conclusion et perspectives

Dans ce rapport, nous avons pu dresser le bilan complet de notre travail qui se situe dans le cadre de notre projet de fin d'études. Ce travail a consisté à concevoir et à réaliser un système de détection des véhicules et de la reconnaissance des plaques d'immatriculation.

Pour y parvenir, nous avons effectué des recherches sur les algorithmes de l'apprentissage supervisé plus précisément l'algorithme des k plus proches voisins, le couplage entre les langages de programmations comme le cas dans notre projet que nous avons couplé Java avec Python et adopter une nouvelle approche de détection des plaques d'immatriculation des véhicules,

Notre application est aujourd'hui achevée et répond à tous les besoins initialement énoncés.

Nous pouvons, cependant suggérer certaines améliorations et extensions possibles. Ajoutons aussi que nous comptons développer la version mobile de cette application pour faire consulter tout l'historique détaillé des véhicules.

Bibliographie

- [1] H. a. B. A. C. a. M. M. a. M. J. Zhang, «KNN: Discriminative nearest neighbor classification for visual category recognition,» chez *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, IEEE, 2006, pp. 2126-2136.
- [2] L. Robert, *OpenCV Computer Vision Application Programming Cookbook Second Edition*, Packt Publishing Ltd, 2014.
- [3] T. Preston-Werner, «GitHub,» 2008. [En ligne]. Available: www.github.com.
- [4] A. a. K. A. Schmidt, «The performance of the haar cascade classifiers applied to the face and eyes detection,» chez *Computer Recognition Systems 2*, Springer, 2007, pp. 816-823.
- [5] U. ObjectAid, *Explorer for Eclipse*, 2014.