“Plates”
Notice de projet

Auteurs	David Darmanger Owen Gombas
Présenté à	Olivier Hüsser
Nom de l’école	Haute École Arc Neuchâtel
Cours	Traitement d’images II
Classe	ISC3id-a
Date de remise	08.05.2023

haute école arc ingénierie
neuchâtel berne jura www.he-arc.ch

1. Introduction

Dans le cadre de notre formation, et plus précisément lors de notre cours de traitement d'images, nous avons réalisé le projet "Plates". Ce projet consiste à développer une application qui met en œuvre des algorithmes et des méthodes de traitement d'images afin de résoudre un problème spécifique. L'objectif de cette application est de détecter les numéros d'identification de plaques d'immatriculation de véhicules à partir de photos prises par une caméra de surveillance. Ces images sont capturées toujours sous le même angle par la même caméra, ce qui facilite l'implémentation de l'algorithme. De plus, étant donné qu'il s'agit de plaques d'immatriculation vietnamienne, le traitement sera adapté aux spécificités de ces plaques.

Pour extraire les caractères contenus sur les plaques d'immatriculation, nous avons mis en place plusieurs algorithmes de traitement d'images tels que Canny, la détection de composantes connexes, et d'autres techniques de segmentation d'images. L'application a été développée en Python, et nous avons utilisé la bibliothèque OpenCV ainsi qu'un modèle de Transformers[1] pré-entraîné. Les images proviennent d'un jeu de données issu d'un projet sur GitHub[2].

Nous avons également créé une interface utilisateur graphique (GUI) conviviale permettant aux utilisateurs de naviguer à travers une galerie d'images, de modifier les paramètres de détection, de lancer le processus de détection des plaques d'immatriculation et d'afficher les résultats en détaillant les différentes étapes. En outre, la précision du score obtenu par le modèle est également affichée.

2. État de l'art et références

Les projets constituant l'état de l'art utilisent essentiellement des algorithmes de Machine Learning afin de procéder à la partie de traitement d'image. Bien que ceux-ci soient extrêmement efficaces, ils sont très abstraits et peuvent demander plus de ressource que des algorithmes de traitement d'image classiques. Cependant, pour la lecture des caractères, ceux-ci nous ont été très utiles, car l'OCR fonctionne principalement sur des principes de Machine Learning.

NNDAM, *Vietnamese-License-Plate-Generator*
https://github.com/NNDam/Vietnamese-License-Plate-Generator

Winter2897, *Real-time-Auto-License-Plate-Recognition-with-Jetson-Nano*
https://github.com/winter2897/Real-time-Auto-License-Plate-Recognition-with-Jetson-Nano

Microsoft, *trocr-base-printed*
https://huggingface.co/microsoft/trocr-base-printed

Symisc, *sod*
https://github.com/symisc/sod
https://sod.pixlab.io/articles/license-plate-detection.html

[1] "microsoft/trocr-base-printed"
[2] "Real time Auto License Plate Recognition with Jetson Nano"

3. Pipeline complet

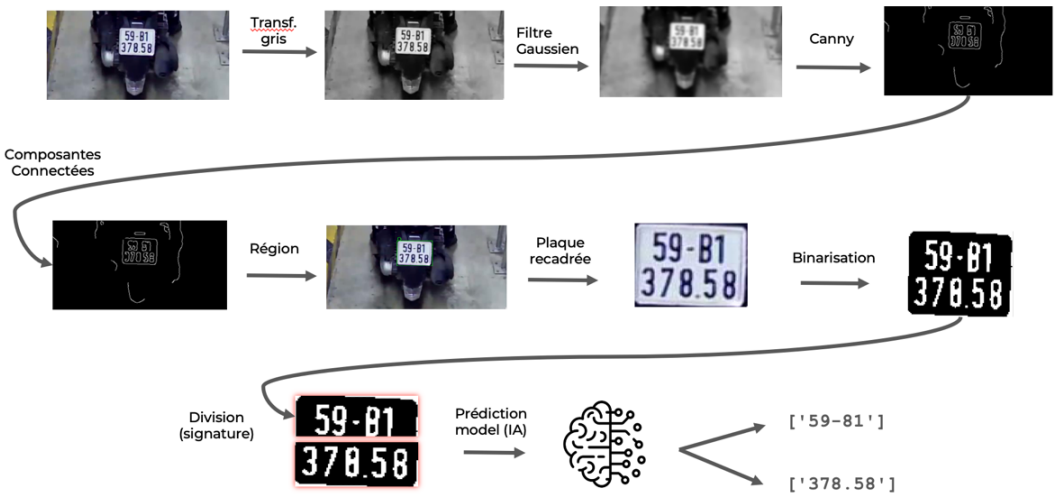

Figure 1 – Pipeline complet de la prédiction de texte

- Importation des images depuis le dossier choisi par l'utilisateur, puis transformation en niveau de gris afin de traiter un seul canal de couleur au lieu de trois.
- Application d'un filtre gaussien pour réduire le bruit présent dans l'image.
- Utilisation de l'algorithme de Canny pour détecter les contours des objets présents dans l'image.
- Analyse des contours pour identifier les composants connectés (même forme ou au même objet).
- Filtrage des composants connectés pour éliminer les régions qui ne sont pas susceptibles de contenir des plaques d'immatriculation, en se basant sur des critères tels que la taille et la forme (rectangle).
- Découpage des régions potentielles à partir de l'image d'origine, puis conversion de chaque région en image binaire afin de faciliter la reconnaissance des caractères.
- Division des régions potentielles en partie supérieure et inférieure pour traiter les numéros d'identification de la plaque séparément.
- Détection des caractères de chaque partie de la plaque à l'aide d'un modèle de Machine Learning pré-entraîné. Le texte détecté est ensuite affiché à l'utilisateur, ainsi qu'un score de précision pour chaque prédiction.

Processus détaillé

Au cours de ce projet, notre principal objectif a été de mettre en place une détection de plaque d'immatriculation fonctionnelle en utilisant des algorithmes de traitement d'image classiques, tout en minimisant l'utilisation de l'apprentissage automatique (Machine Learning). Pour commencer, nous rognons l'image de N pixels sur chaque côté afin d'enlever de l'informations inutiles constituant les bords de l'image (paramètre CROP= [10, 10, 10, 10]). Ensuite il a été nécessaire de mettre en place une détection de contours sur l'image. Après avoir effectué une série de tests avec les différentes méthodes vues en classe (dérivées, Prewitt, Sobel, Canny), nous avons constaté que la méthode de Canny est la plus efficace. Cependant, pour obtenir les meilleurs résultats possibles, il est nécessaire d'appliquer un filtre gaussien au préalable pour supprimer le bruit de l'image, la taille du kernel peut être modifiée dans les paramètres du programme afin de perfectionner la détection sur une image spécifique (GAUSSIAN_KERNEL= [15, 15], fonctionne bien de façon générale).

Après avoir obtenu les contours, nous pouvons extraire de cette image toutes les composantes connexes grâce à OpenCV, cette méthode extrait bien la plaque, mais on récupère également énormément de régions n'incluant aucune plaque. Pour pallier à ce problème, nous avons appliqué des règles que chacune des régions doit respecter pour être considérée comme des plaques d'immatriculation : l'« aspect ratio », la longueur et la hauteur doivent être inclus dans un intervalle (qui peut être changé via les paramètres).

- ASPECT_RATIO_MIN_MAX= [0.3, 1.5]
- HEIGHT_MIN_MAX= [40, 150]
- WIDTH_ MIN_MAX= [40, 140]

De plus, on peut spécifier la taille minimale des composantes connexes afin de filtrer celles qui sont trop petites (MIN_SIZE_CONNECTED_COMPONENT= [10]). Une fois les régions contenant potentiellement des plaques extraites, nous pouvons mettre en évidence et extraire cette partie de l'image originale. Cependant, malgré cela, il peut arriver que le processus de traitement d'image sélectionne de mauvaises régions qui ne sont pas des plaques, ce qui constitue une des limitations de notre méthode.

Enfin, chaque région potentiellement détectée comme plaque d'immatriculation est convertie en image binaire pour faciliter la reconnaissance des caractères. Ensuite, nous devons les découper en deux parties afin d'isoler chacune des lignes de texte. Ce processus est obligatoire, car le modèle d'intelligence artificielle utilisé ne prend en charge que le texte sur une seule ligne. Pour ce faire, nous avons mis en place une technique de séparation en deux parties supérieure et inférieure basée sur une « signature ».

Cette technique consiste à rechercher la ligne qui contient le plus de pixels noirs (si possible sur toute la ligne), et une fois trouvés, à la symboliser comme étant la séparation entre les deux lignes. Cependant, il est important de prendre en compte que les bords de l'image peuvent contenir du bruit qui pourrait fausser la solution de la séparation. Pour éviter cela, nous avons exclu les bords de l'image de notre recherche de la ligne de séparation, en nous limitant à une zone centrale de l'image.

Cette méthode a permis d'obtenir une séparation de la plaque en deux parties distinctes de façon efficace et surtout adaptable aux différentes images dans laquelle les plaques ne sont pas toujours de la même taille, ou dans le même angle.

Les caractères sont ensuite détectés séparément sur chaque division de plaque en utilisant un modèle de Transformers pré-entraîné et la bibliothèque OpenCV. Le texte détecté est ensuite affiché à l'utilisateur, ainsi qu'un score de justesse pour chaque prédiction.

4. Résultats

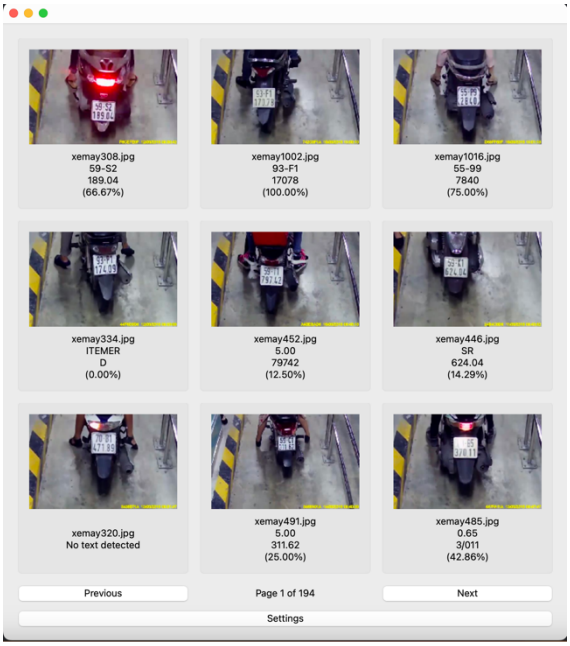

Figure 2 - Page d'accueil / Galerie d'images

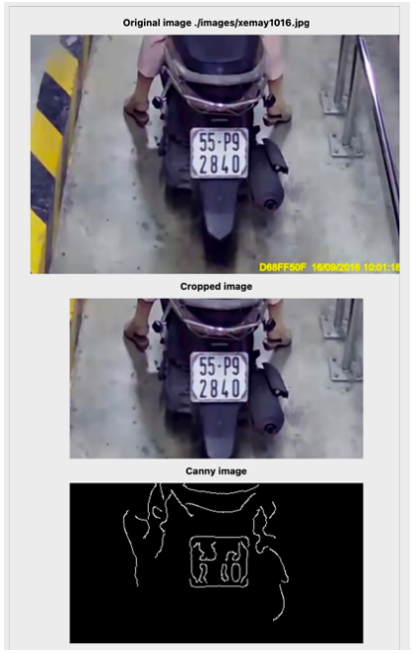

Figure 3 - Pipeline de traitement d'image (Canny)

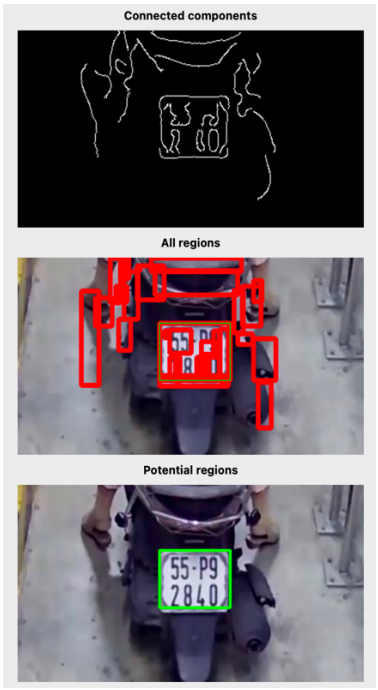

Figure 4 - Pipeline de traitement d'image (Détection de plaques potentielles)

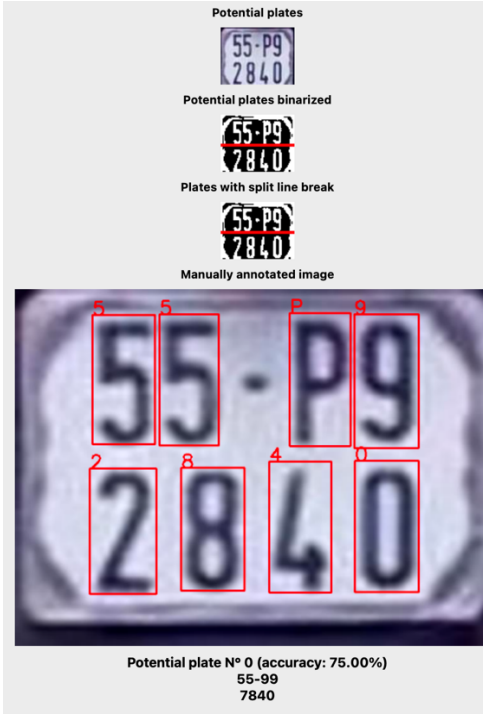

Figure 5 - Pipeline de traitement d'image (Détection de texte)

5. Discussions des résultats

Les résultats obtenus par notre application peuvent varier selon les images d'entrée. Il peut parfois y avoir des erreurs de détection, comme une plaque non détectée ou une portion de l'image identifiée comme une plaque alors qu'elle ne l'est pas. Toutefois, malgré ces problèmes de détection, les résultats globaux sont plutôt satisfaisants et convaincants pour ce qu'il en est de la détection des plaques. En ce qui concerne la conversion des caractères de l'image en texte, nous obtenons un score de prédiction qui peut être améliorée si nous avions eu plus de temps et de ressource de calcul. Par exemple en entraînant un modèle spécifiquement sur nos plaques d'immatriculation et nos labels. Cependant la majorité des caractères que nous pouvons percevoir clairement sont reconnus et convertis en texte de façon efficace. Le score de conversion obtenu varie beaucoup d'une image à une autre (haute variance), il est donc peu pertinent de donner un score de prédiction final en établissant une moyenne pour cette première version du modèle.

En effet, dans la plupart des cas, la totalité des caractères présents sur les plaques d'immatriculation est extraite avec précision. On constate souvent seulement une ou deux erreurs de caractères. Ces erreurs peuvent être dues à des conditions d'éclairage difficiles ou à des caractères mal alignés sur la plaque. Malgré ces erreurs, l'application reste une solution efficace pour la détection et la reconnaissance des plaques d'immatriculation.

6. Limitations et problèmes

Nous avons constaté que le processus de traitement d'image peut parfois sélectionner de mauvais endroits en forme de rectangle qui ne sont pas des plaques, ce qui peut entraîner une détection incorrecte. Ce problème peut être attribué à la complexité de l'environnement et à la présence d'objets similaires à des plaques d'immatriculation. Nous avons également remarqué que, dans certaines situations, le modèle ne détecte pas les plaques présentes sur l'image, même s'il en existe une. Enfin, le modèle peut parfois se tromper sur quelques caractères détectés, même si les résultats globaux restent satisfaisants. Pour cela nous avons ajouté une section « Settings » permettant de modifier les paramètres de détection de plaque dans notre application, ce qui permet d'améliorer la flexibilité de détection de plaque sans utiliser de Machine Learning.

En termes de limitations, notre pipeline est conçu pour des images provenant d'une même caméra avec un angle de vue similaire, et pour des plaques d'immatriculation vietnamienne uniquement. Cela limite son évolutivité à d'autres situations. Cependant, il est facilement adaptable en modifiant les paramètres pour s'adapter à de nouveaux problèmes. En ce qui concerne les limitations liées au modèle, il est capable de détecter uniquement les textes sur une seule ligne, ce qui nous a obligés à séparer la plaque d'immatriculation en deux parties en utilisant un concept de "signature" assez basique. Ce processus de séparation pourrait être amélioré à l'avenir en utilisant des techniques plus avancées.