

Reconnaissance de l'écriture manuscrite

Rapport de projet

Laurent Antoinette, Romain Campillo, Tony Nguyen
L3 informatique
Faculté des Sciences
Université de Montpellier.

10 mai 2023



Table des matières

1	Présentation du Sujet	4
1.1	Le problème	4
1.2	Les différentes approches	4
1.2.1	k-Nearest Neighbor	4
1.2.2	Template matching	4
1.3	Objectifs	4
1.4	Cahier des charges	4
1.4.1	Besoins et contraintes	4
1.4.2	Résultats attendus	5
2	Technologies utilisées	5
2.1	Langages et outils	5
2.1.1	Python3	5
2.1.2	OpenCV	5
2.1.3	You Only Look Once	5
3	Développements Logiciel : Conception, Modélisation, Implémentation	7
3.1	Interface Homme-Machine	7
3.2	Procédures de lecture et validation des entrées	8
3.3	Statistiques	8
4	Algorithmes et Analyse	8
4.1	YOLO	8
4.2	Segmentation par histogramme de projection	8
5	Analyse des Résultats	13
6	Gestion du Projet	13
	Bilan et Conclusions	13

Table des figures

1	Diagramme de cas d'utilisation	7
2	Fenêtre de l'IHM quand on modifie une image	7
3	Fenêtre de l'IHM après présentation des caractères reconnus	8
4	Exemple d'une image après filtrage	10
6	Structure de données dans l'algorithme SEGMENTATION	10
5	Images de nos 3 lignes crée à partir de l'image de départ	10
7	Imagettes des caractères crée à partir de la 1e ligne	12
8	Imagettes des caractères crée à partir de la 2e ligne	12
9	Extrait d'un histogramme de projection horizontal	15

List of Algorithms

1	SEGMENTATION(image)	9
2	COORDONNEECARACTERE(T)	11

1 Présentation du Sujet

1.1 Le problème

L'essence du problème est qu'un texte dans un document est très différent d'une photo du même document. Dans le 1er cas, le format rend le contenu compréhensible par un ordinateur, il peut donc chercher un mot dans le texte ou le modifier pour corriger des fautes d'orthographe. Dans le 2e cas, l'image représente la même chose. MAIS, la machine ne comprend le sens de l'information représentée. Il voit les pixels mais pas les caractères. De plus, des imperfections dans la capture du document changent la photo (ex : luminosité, angle ...), il y a une modification des informations alors que le texte ne change pas.

Réussir à lire du texte sur des images permettrait de faciliter la numérisation de document papier. Ainsi, libérer les Humains de quelques tâches redondantes. On peut imaginer ce genre de système en train de traiter des formulaires ou encore d'améliorer les moteurs de recherche avec des images qui contiennent du texte.

Ce problème est particulièrement intéressant. En effet, tout ce qui traite de l'information appartient à notre domaine d'étude. En plus de l'automatisation. Nous avons pu constater une explosion de l'intelligence artificielle ces dernières années. Notamment les réseaux neuronaux. La reconnaissance d'image fait partie des objectifs qu'ils ont déjà atteints. Ce projet est tout à fait intéressant pour de futurs étudiants en master IASD.

1.2 Les différentes approches

1.2.1 k-Nearest Neighbor

Méthode statistique, il y a un plan avec plein d'objets qui forment des troupes, notre lettre est quelque part dans le plan, on trace un cercle autour, notre lettre est identifiée comme l'objet présent au plus grand nombre à l'intérieur du cercle.

1.2.2 Template matching

En gros, comparaison de l'image avec la base de données entière à l'aide d'une fonction de distance.

1.3 Objectifs

Notre but est de créer un logiciel dont une personne se servirait pour transformer une image qui contient des symboles en un texte manipulable par un ordinateur.

Notre but initial sera de reconnaître des images simples et de bonne qualité avec des symboles de l'alphabet latin (26 lettres) seulement en majuscules sur 1 ligne.

1.4 Cahier des charges

1.4.1 Besoins et contraintes

Les besoins

Capter une image L'utilisateur pourra prendre des photos par notre logiciel à l'aide d'une webcam. Mais il pourra également utiliser des images depuis son système de fichiers. Tout cela à travers une interface Homme-Machine.

Pré-traitement Le logiciel réduira le bruit et rendra l'image plus nette à l'aide de plusieurs filtres prédéfinis.

Segmentation de l'image Les différents caractères présents sur l'image seront localisés à l'aide d'un histogramme de projection.

Extraction des caractères Les caractères seront ensuite découpés pour former leurs propres imagerie .i.e une image qui contient TEST deviendra 4 petites images contenant respectivement T E S T.

Reconnaissance Les images feront l'objet d'une reconnaissance de façon individuelle. Dans notre cas nous choisissons d'utiliser un réseau neuronal convolutif.

Présenter Une fois que les images sont reconnues en caractère. Il est nécessaire de les assembler et d'afficher à l'utilisateur les mots reconnus.

Les contraintes Nous nous fixons comme contraintes de ne pas utiliser de services comme Google Collab car Google possède un modèle économique type "Freemium" (gratuit mais payant). Nous souhaitons créer un logiciel suffisamment performant pour pouvoir être exécuter sur l'une de nos machines personnelles.

De plus, une webcam est nécessaire, ou alors un appareil photo numérique.

1.4.2 Résultats attendus

Notre programme doit reconnaître des lettres manuscrites sur un fond blanc. Les lettres seront des caractères majuscule non-liées et sans accents ni caractère spécial.

2 Technologies utilisées

2.1 Langages et outils

2.1.1 Python3

2.1.2 OpenCV

Dans ce projet, nous choisissons d'utiliser la librairie open source de vision par ordinateur (Open Computer Vision). Elle contient les fonctionnalités de pré-traitement d'image nécessaire dans ce projet. Nous aurons notamment besoins des fonctions de filtrage d'image.



2.1.3 You Only Look Once

You Only Look Once (YOLO) est une architecture de Réseau Neuronal Convolutif (CNN) de type "Region based" (R-CNN) qui est capable de localiser des objets dans une image et en même temps, les classifier.

Expliquons (un peu tout petit peu) plus en détail les réseaux neuronaux

Les réseaux de neurone simulent plus ou moins le fonctionnement de notre cerveau. Parlons d'abord de l'élément le plus petit et on va progresser jusqu'aux CNN.

Le neurone va prendre plusieurs entrées, des nombres, faire une opération avec une fonction d'activation et en sortie retourner le résultat.

Réseau de neurone L'architecture des réseaux de neurone simple est organisée en couche. La 1er couche qui est l'entrée (censée représenter les yeux). La dernière couche représente la sortie, par exemple si le réseau est entraîné pour reconnaître des chiens, la sortie doit retourner une valeur proche de 1 si c'est un chien et 0 sinon. Entre le début et la fin du réseau, les couches intermédiaires sont connectées vers l'avant et de façon complète. Un neurone est connecté à tous les neurones de la couche précédente et suivante mais pas à ceux de la même couche.

Le désavantages de cette approche est qu'il est nécessaire de lui donner des "caractéristiques". Pour

reconnaitre si cet animal est un chien on ne va pas lui donner l'image directement, on va lui dire en entrée si ça a des poils, la couleur de son pelage, le nombre de pattes, la forme des oreilles.

Les CNN blablabla

R-CNN

test

Qu'est que c'est?. C'est une phrase français avant le lorem ipsum. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

3 Développements Logiciel : Conception, Modélisation, Implémentation

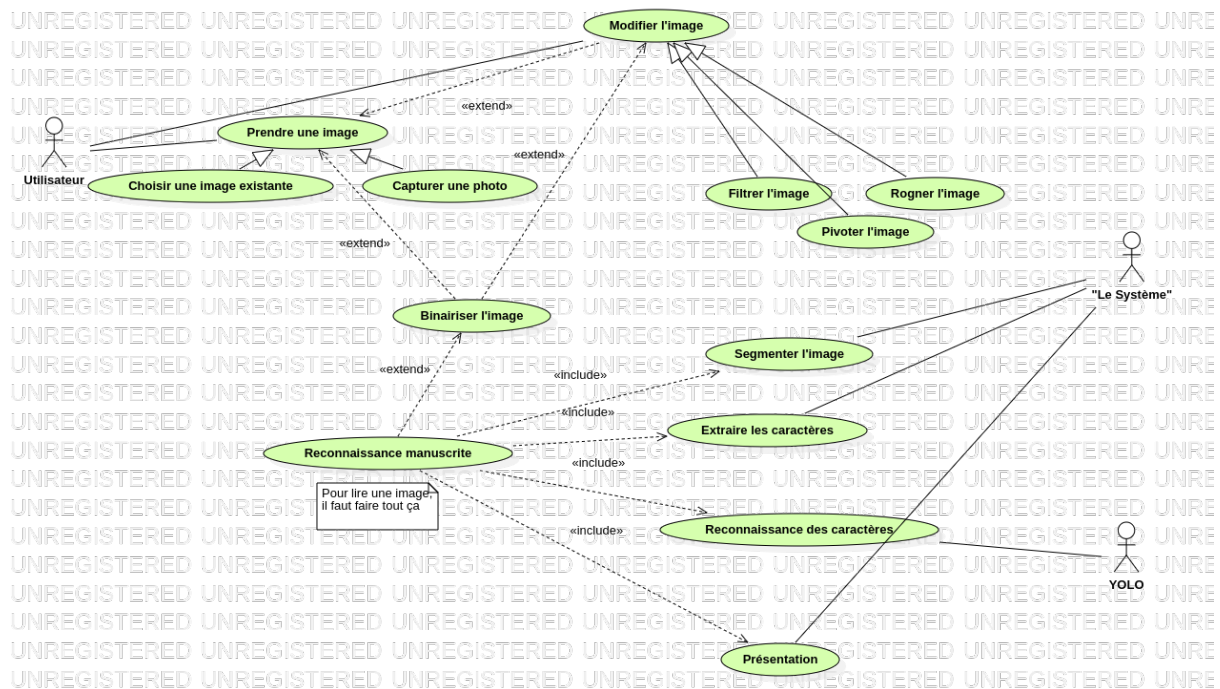


FIGURE 1 – Diagramme de cas d'utilisation

3.1 Interface Homme-Machine

L'IHM nous permet de prendre des photos avec la webcam connectée à l'ordinateur ou choisir à partir d'une fenêtre un chemin vers une image.

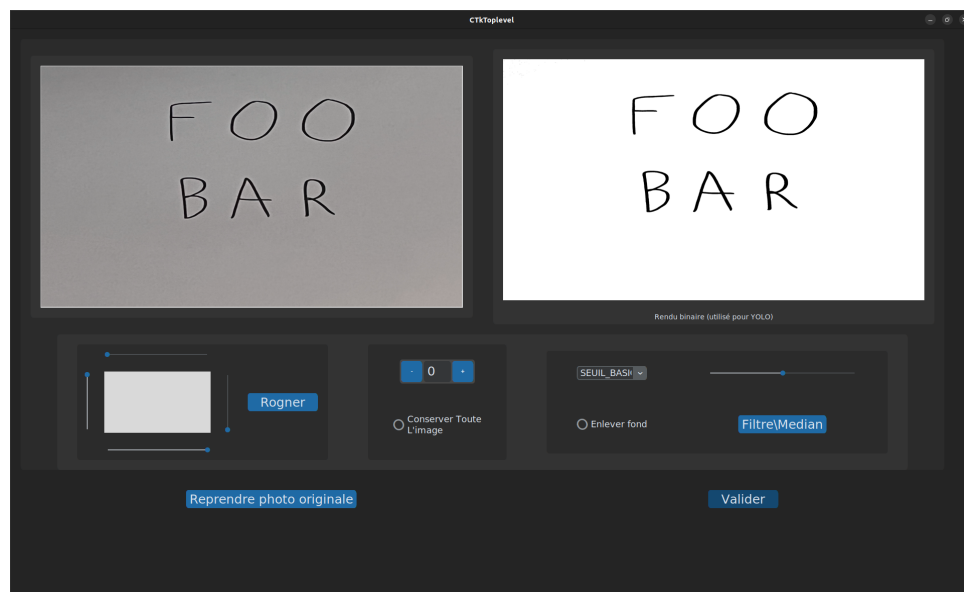


FIGURE 2 – Fenêtre de l'IHM quand on modifie une image

Après avoir entré une image, il est possible qu'il y ait beaucoup de bruit, dans ce cas là, nous pouvons rogner l'image pour enlever des ombres sur les bords, faire des rotations et filtrer pour enlever le bruit comme illustrée sur la figure 2



FIGURE 3 – Fenêtre de l'IHM après présentation des caractères reconnus

Sur la figure 3, on peut voir qu'après avoir obtenu une image de bonne qualité, notre programme va segmenter les caractères, en faire des imagerie et donner ces imagerie individuellement à YOLO. Ensuite après cette reconnaissance, l'IHM présente le résultat.

Il est possible de voir les imagerie générées dans le dossier `"/images/imageSegmentee"`

3.2 Procédures de lecture et validation des entrées

beep

3.3 Statistiques

- Nombres de lignes de code :
- Nombres de script

4 Algorithmes et Analyse

4.1 YOLO

4.2 Segmentation par histogramme de projection

Algorithm 1: SEGMENTATION(image)

Data: image en RGB

Stratégie: D'abord, une segmentation des lignes par projection vertical. Puis, dans un 2e temps, segmentation des caractères grâce à une projection horizontal de chaque ligne individuellement

Result:

Tableau de couples d'entier représentant les coordonnées y de début et de fin des lignes sur l'image

Tableau de tableau de couple d'entier représentant les coordonnée x de début et de fin des lettres par lignes,

Tableau d'image des lignes

Tableau de tableau d'image des caractères

```
1 ndgImage ← convertirImageEnNDG(image);
   /* on met l'image en noir et blanc sauf que le seuillage est inversé */
2 imageBinaire ← binarisationInvers(ndgImage);
   /* on fait la somme des pixels sur les colonnes et on les divise par 255 */
3 projectionHorizontale = (sommeValPixel(imageBinaire,axe = 1))/255;
4 delimitationDesLignes = coordonneeLigne(projectionHorizontale);
5 imagesLignes = [];
   /* on copie une zone de l'image correspondant à un caractère grâce aux
      coordonnées calculées */
   /* et on l'ajoute à notre tableau d'image des lignes */
6 for (x,y) dans delimitationDesLignes do
7   | imagesLignes.ajouter(ndgImage[x :y, 0 :longueurImage])
8 end
   /* on initialise le tableau de tableaux des images de Caractère on fonction du
      nombre de ligne detecté */
9 imagesCaracteres = [[]pouriallantde0longueur(imagesLignes)];
   /* on initialise le tableau de tableaux de couple de coordonnées on fonction du
      nombre de ligne */
10 coordonneesCaracteres = [[]pouriallantde0longueur(imagesLignes)];
11 for index, uneLigne dans imagesLignes do
12   | ligneBinaire = binarisationInvers(uneLigne);
      /* on fait la somme des pixels sur les lignes et on les divise par 255 */
13   | projectionVerticale = (sommeValPixel(imageBinaire,axe = 0))/255;
14   | delimitationDesCaracteres = coordonneeCaractere(projectionVerticale);
      /* on ajoute les coordonnées au tableau de coordonnées */
15   | coordonneeCaractere[index].ajouter(delimitationDesCaracteres);
      /* on copie une zone de l'image correspondant à un caractère grâce aux
         coordonnées calculées */
      /* et on l'ajoute à notre tableau d'image de caractère */
16   | for (x,y) dans delimitationDesCaracteres do
17     | imagesCaracteres[index].ajouter(ndgImage[0 : hauteurUneLigne, x : y]);
18   | end
19   | redimensionnerImage(imagesCaracteres[index], hauteur = 128, longueur = 128);
20 end
21 return delimitationDesLignes, delimitationDesCaracteres, imagesLignes, imagesCaracteres
```



FIGURE 4 – Exemple d’une image après filtrage

Sur la figure 4, nous voyons une image de bonne qualité prête à être segmenter par l’algorithme n°1 (page 9).

Grâce à une projection vertical, nous obtenons ce tableau (voir figure 6). La 1er ligne de ce tableau contient autant de tuples que le nombre de lignes. Chaque tuples contient la composante en ordonnée du début et de la fin d’une ligne. En choisissant arbitrairement comme largeur d’une ligne, la largeur de l’image, on extrait nos 2 lignes comme sur la figure 5.

```
0., 0.])
coord_ligne=[(96, 224), (330, 454)]
coord_caracteres=[(337, 465), (522, 664), (739, 900)]
coord_caracteres=[(368, 451), (522, 647), (744, 840)]
[]
```

FIGURE 6 – Structure de données dans l’algorithme SEGMENTATION



FIGURE 5 – Images de nos 3 lignes crée à partir de l’image de départ

Algorithm 2: COORDONNEECARACTERE(T)

Data: tableau de flottant représentant la projection vertical d'une image

Stratégie: Trouver les zones non nul dans l'histogramme

Result: tableau de couple d'entier indiquant les coordonnées du début et de la fin d'une lettre

```
1 coordCaractere = [];  
2 dansLeCaractere = False;  
  /* Début = pair Fin = impair */  
3 DF = [];  
4 for i de 0 à taille(T) do  
5   if T[i] ≠ 0 then  
6     if !dansLeCaractere or (dansLeCaractere and i == taille(T)-1) then  
7       DF.ajouter(i);  
8     end  
9     dansLeCaractere = True;  
10  end  
11  else  
12    if dansLeCaractere then  
13      DF.ajouter(i)  
14    end  
15    dansLeCaractere = False;  
16  end  
17 end  
18 lettres = [(DF[i], DF[i + 1]) pour i de 0 à taille(DF) avec un pas de 2];  
19 tailleEspacesEntrelettres = [lettres[i + 1][0] - lettres[i][1] pour i de 0 à taille(lettres) - 1];  
20 poidsEspace = trillageCroissant(tailleEspacesEntrelettres);  
21 SommeIndice ← 0;  
22 for i de 0 à taille(poidsEspace)-1 do  
23   poidsEspace[i] ← poidsEspace[i] * (taille(poidsEspace) - i);  
24   SI ← SI + i;  
25 end  
26 moyenneEspaces ← somme(poidsEspace)/SI;  
27 j ← 0;  
28 while tailleEspacesEntrelettres != [] do  
29   if tailleEspacesEntrelettres < moyenneEspaces * 0.2 then  
30     D = lettres[j]; supprimer(lettres[j]);  
31     F = lettres[j]; supprimer(lettres[j]);  
32     /* inserer(tableau, indice, élément) */  
33     inserer(lettres, j, (D[0], F[1]));  
34   else j ← j + 1;  
35   supprimer(tailleEspacesEntrelettres[0]);  
36 end  
37 return lettres
```



FIGURE 7 – Imagettes des caractères crée à partir de la 1e ligne

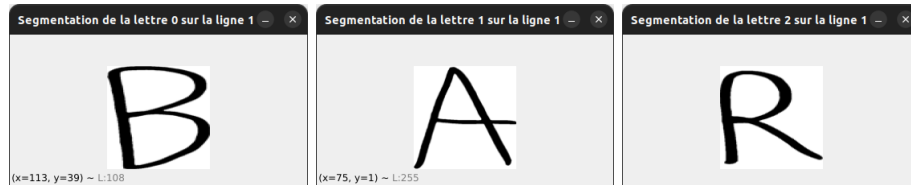


FIGURE 8 – Imagettes des caractères crée à partir de la 2e ligne

Maintenant que nous avons isoler les lignes, nous allons maintenant découper les lettres de chaque lignes en suivant l'algorithme n°2 page 11

Étant donnée que l'image est filtré, en noir et blanc et binairisé, lorsqu'une lettre est présente à un endroit donnée, cela signifie que sur la projection vertical, l'histogramme est supérieur à 0 à cet endroit là. Sinon, il n'y a pas de lettre et l'histogramme est à 0. L'algorithme n°2 va parcourir totalement l'histogramme de projection et va tenté de dectecter ces "zones pleines" et récupérer l'indice de début et de fin de ces zones qui correspondront à la limite gauche et droit de la lettre sur l'image. Nous pouvons voir le résultats de cette fonction sur la figure 6 et il y a extrait d'un histogramme horizontal dans l'annexe (page 15).

5 Analyse des Résultats

TBD

6 Gestion du Projet

Pendant ce projet, nous avons dû communiquer et collaborer ensemble afin d'atteindre notre but commun. Voici les outils que nous avons utilisés.

Premièrement, pour communiquer nous avons utilisé l'application Discord. Sur lequel nous avons un groupe où nous nous envoyons des messages, des images et des fichiers.

Ensuite, pour le partage et le versioning, nous utilisons git et Github. Laurent : filtrage, segmentation
Romain : IHM, YOLO Tony : rapport

Bilan et Conclusions

tests

Cependant, certains points laissent à désirer dans notre projet. Malheureusement, une contribution de l'utilisateur est nécessaire. Pour une bonne reconnaissance des caractères, l'utilisateur doit rogner l'image. Il serait plus pratique que le *rognage soit fait automatiquement*.

Ensuite, au niveau de la *segmentation*, notre algorithme n'est pas capable de séparer des lettres cursives et ignore totalement les espaces.

Et enfin, dans le futur, le but sera que *YOLO* reconnaisse aussi l'alphabet minuscule mais aussi l'alphabet français (avec les accents).

Bibliographie

TBD

Annexes

[illegible]

FIGURE 9 – Extrait d'un histogramme de projection horizontal