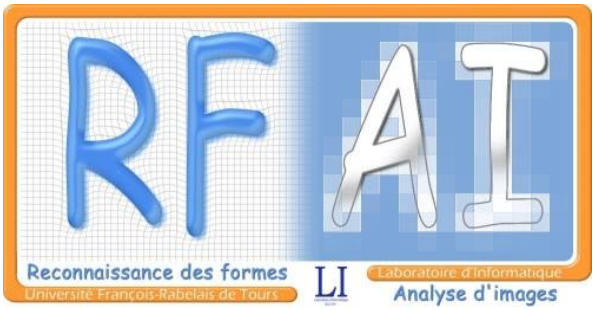
Ecole Polytechnique de l'Université François Rabelais de Tours
Département Informatique
64 avenue Jean Portalis
37200 Tours, France
Tél. +33 (0)2 47 36 14 14
www.polytech.univ-tours.fr

Projet Recherche & Développement
2015-2016

Plateforme pour word spotting multi-scripts

Plateforme pour word spotting multi-scripts



Entreprise
Laboratoire Informatique

Tuteurs entreprise

Étudiants
Yamin Zaidou (DI5)

Tuteurs académiques
Nicolas Ragot

24 mars 2016

Liste des intervenants

Entreprise

Laboratoire Informatique 64 avenue Jean Portalis, 37200 Tours li.univ-tours.fr

Nom	Mail	Qualité
Yamin Zaidou	yamin.zaidou@etu.univ-tours.fr	Étudiant DI5
Nicolas Ragot	nicolas.ragot@univ-tours.fr	Tuteur académique
		Tuteur entreprise

Avertissement

Ce document a été rédigé par Yamin Zaidou susnommé l'auteur.

L'entreprise Laboratoire Informatique est représentée par susnommé le tuteur entreprise.

L'école polytechnique de l'université François Rabelais de Tours est représentée par Nicolas Ragot susnommé le tuteur académique.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

L'auteur reconnait assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

L'auteur atteste que les propos du document sont sincères et assument l'entière responsabilité de la véracité des propos.

L'auteur atteste ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

L'auteur atteste que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

L'auteur reconnait qu'il ne peut diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable du tuteur académique et de l'entreprise.

L'auteur autorise l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.

Pour citer ce document :
Yamin Zaidou, *Plateforme pour word spotting multi-scripts: Plateforme pour word spotting multi-scripts*, Projet Recherche & Développement, Ecole Polytechnique de l'Université François Rabelais de Tours, Tours, France, 2015-2016.

```
@mastersthesis{
    author={Zaidou, Yamin},
    title={Plateforme pour word spotting multi-scripts: Plateforme pour word spotting multi-scripts},
    type={Projet Recherche \& Développement},
    school={Ecole Polytechnique de l'Université François Rabelais de Tours},
    address={Tours, France},
    year={2015-2016}
}
```

Document mis en forme sous LaTeX
grâce à la classe de document `polytech.cls` V1.0 (2016/02/01)

Table des matières

Table des figures

Liste des tableaux

1 Introduction

Ce projet de recherche et développement concernent l'évolution d'une plate-forme de recherche par mot-image dans des collections d'images et est réalisé dans le cadre de la formation d'ingénieur à l'École polytechnique de l'université Tours. Il est encadré par M. Nicolas RAGOT et s'inscrit dans la continuité de trois PFEs des années précédentes réalisés respectivement par Zheng ZHANG, Dawei SHEN et principalement du dernier, réalisé par Loreen LAMBIN qui a pu complètement intégrer un outil de spotting (le projet renom) implémenté par M. Frédéric RAYAR, et avait commencé à intégrer un deuxième outil de CDP du PFE de Bastien MEUNIER.

1 Contexte et problématique

Depuis plusieurs années, de grandes quantités de documents d'archives sont numérisés et mis en ligne pour un accès public et dans une optique de conservation du patrimoine. Ces documents, imprimés où manuscrits, sont souvent dégradés. Le fait que ces documents soient des images il est impossible de rechercher un mot directement à partir des caractères alpha-numérique. La recherche ne peut se faire donc que par feuilletage dans la majeure partie de ces documents. Ce projet s'inscrit ainsi dans un processus pour effectuer une recherche dans des images.

Effectivement à cause des échecs des applications de reconnaissance Optique de Caractères (OCR) sur les documents manuscrits et les vieux documents imprimés, une des solutions pour remédier à ce problème est l'utilisation des techniques de word-spotting, techniques qui permettent d'identifier un paramètre d'entrée, dans notre cas une image, dans le ou les documents souhaités (une collection d'images représentant un livre par exemple).

2 Introduction au principe du word-spotting

La recherche se faisant sur une base d'images, le principe est de récupérer, en premier lieu lors de son importation, toutes les instances possibles des lignes/mots (la segmentation[11]) puis d'en extraire des caractéristiques [17] spécifiques. Lors de la recherche par un utilisateur,le mot recherché sous forme d'image subit également la même approche d'extraction de caractéristique.

Un outil de word-spotting est alors utilisé pour comparer rapidement les caractéristiques de la requête/image et ceux des instances de la base. Les meilleurs résultats sont ensuite affichés à l'utilisateur. Plusieurs traitements peuvent intervenir entre ses différentes actions et seront expliqués dans l'état de l'art [10].

3 Environnement du projet

La plate-forme doit permettre à l'utilisateur de parcourir des images représentant un document, d'effectuer une recherche à partir d'une image clé en ayant la possibilité de choisir l'outil de word-spotting, et de visualiser les résultats de la recherche. Ses différentes fonctionnalités sont décrites dans le cahier de spécification [1] et dans le rapport de Loreen LAMBIN [2]. Pour résumer, voici la nature des principaux échanges :

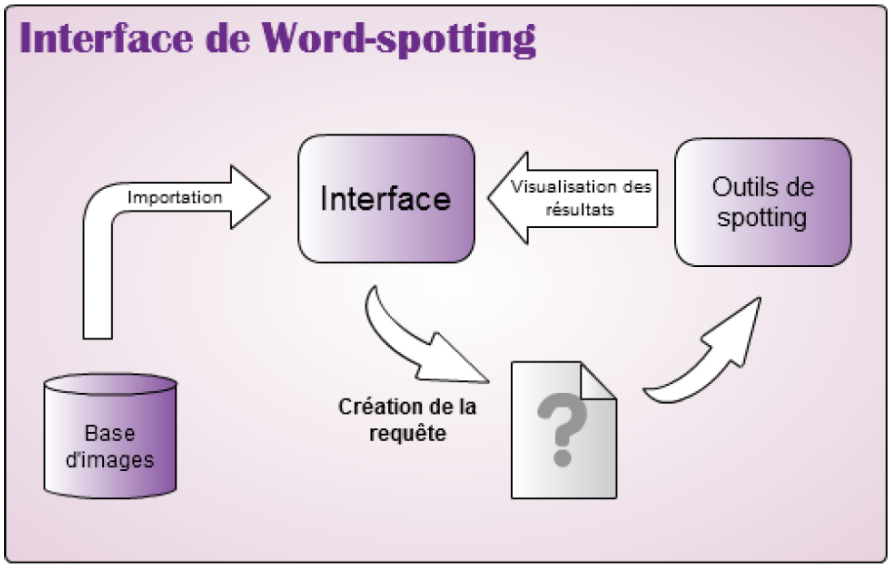


Figure 1 – *Interactions entre l'interface et les outils. source : [2]*

Comme ce PR&D est une reprise de précédent projets, des contraintes de développement et d'environnement devront être respectées. Ainsi les tests et le développement seront réalisés en .NET C#, sous une machine Windows, avec l'IDE Visual studio 2012. Plusieurs librairies externes telles que AForge. NET et OpenCV devront être utilisées pour faire fonctionner les outils correctement. De plus ce PR&D nécessitera l'intégration d'autres programmes développés en C++ (Extracteur de caractéristiques, Segmentation et la toolbox CDP de Bastien MEUNIER).

4 L'existant

Ce projet étant principalement une reprise de celui de Loreen LAMBIN, la version récupérée est fonctionnelle pour l'importation d'une base d'image, la possibilité de rechercher une image avec l'outil Renom un début d'utilisation de l'outil de Bastien MEUNIER, et l'affichage des résultats. Des analyses UML ont été conçues en annexe du PFE de Loreen LAMBIN [2], pour une amélioration de la plate-forme.

4.1 Plate-forme de word-spotting

4.1.1 Interfaces et fonctionnalités

A l'ouverture de l'application la fenêtre ci-dessous s'ouvre.

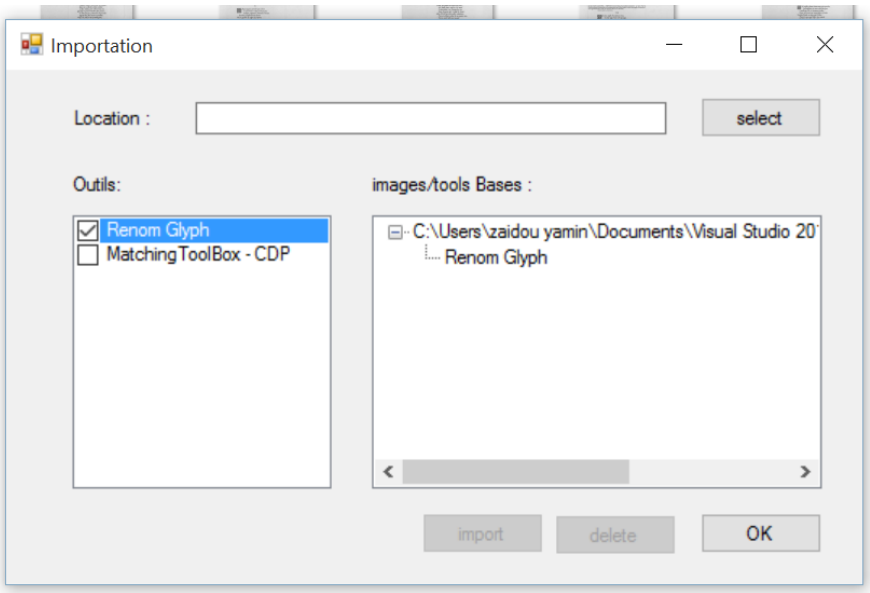


Figure 2 – *Fenêtre Importation*

Cette fenêtre propose à l'utilisateur trois actions différente :

- L'importation d'une base d'image : L'importation consiste à créer dans le dossier de la base (qui doit respecter une structure précise décrite dans le rapport [2]) des vignettes représentant les images mais en plus petite taille. Elle consiste aussi à écrire dans un fichier (BaseOutils.xml) le lien vers la base et les outils de recherches que l'utilisateur a sélectionné.
- L'ouverture/Suppression de la base :
  Une fois la base importée, l'utilisateur peut choisir de l'ouvrir ou de la supprimer. Cette dernière fonctionnalité déclenche la suppression des vignettes dans le dossier et la suppression des liens dans le fichier (BaseOutils.xml).

Une fois une base d'image importée, celle-ci s'ouvre dans la fenêtre suivante : Cette interface permet de

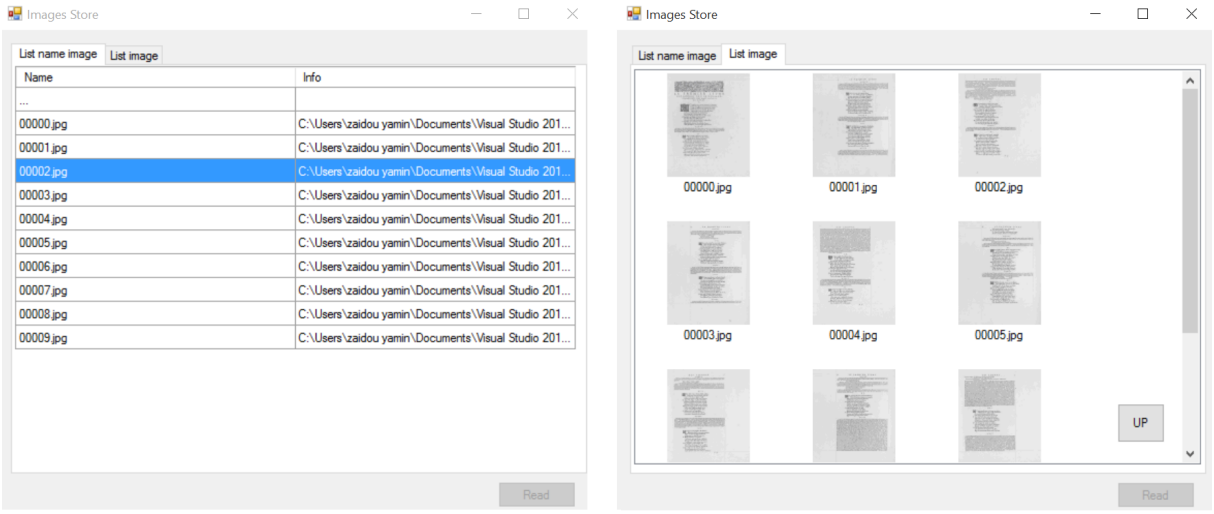


Figure 3 – *Fenêtre informant les images présents dans la base*

visualiser les noms ou les vignettes des images de la base. L'utilisateur peut alors ouvrir une image pour accéder à l'interface de navigation.

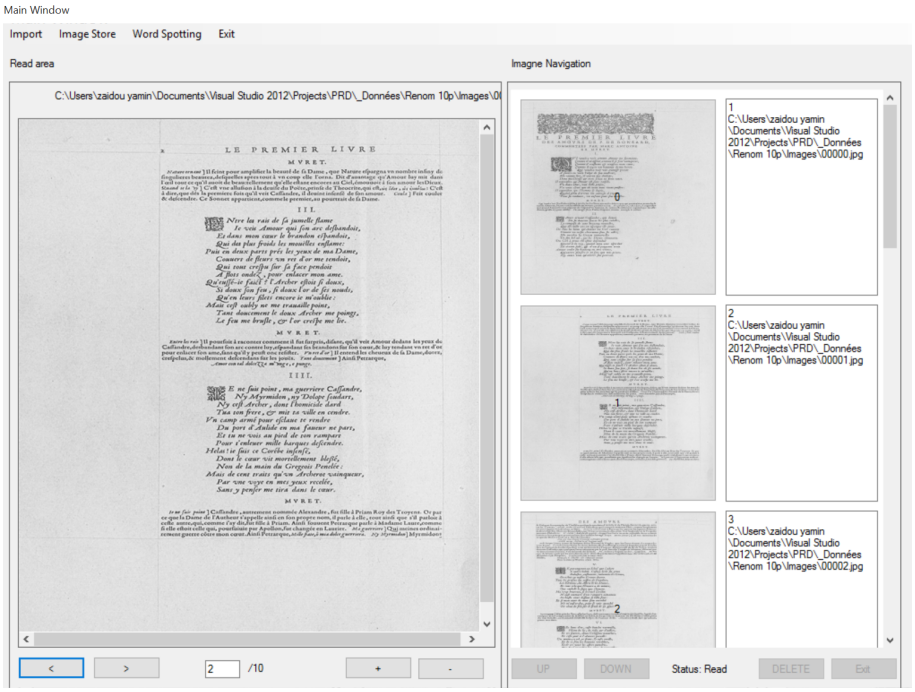


Figure 4 – *Fenêtres de navigation sur la base d'image*

Il s'agit de la fenêtre principale, comme expliqué dans le rapport [2] avec plus de détails, elle permet à l'utilisateur une meilleure navigation entre les différentes images de la base, ainsi que l'affichage des résultats issus de la recherche.

L'application permet d'effectuer, selon les outils de word-spotting, deux types de recherches : par zone de sélection (l'utilisateur à la possibilité de sélectionner une image à rechercher directement sur une image représentant un document) ou en composant des mots à partir d'un dictionnaire proposé :

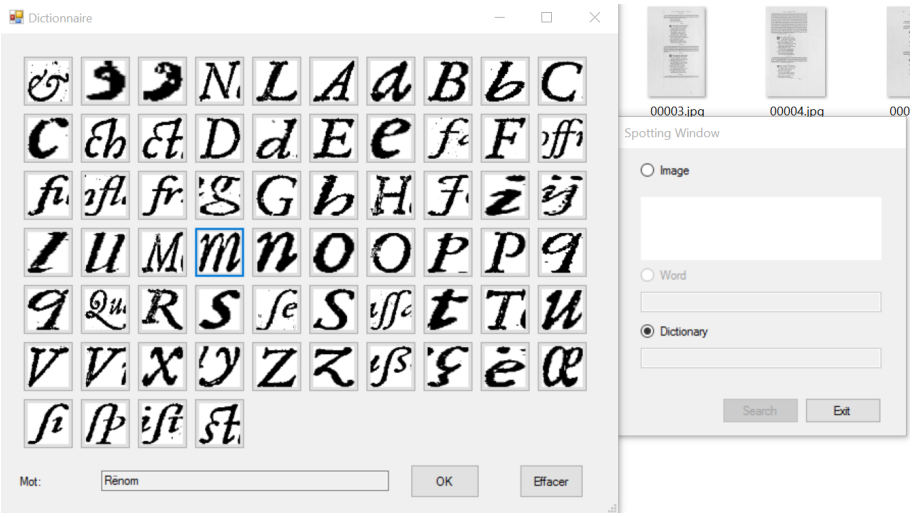


Figure 5 – *Fenêtres de recherche par dictionnaire*

La recherche par dictionnaire permet de composer un mot sans forcément prendre un exemple dans les documents, cette recherche est moins précise et obtient moins de résultats dus parfois au style d'écriture ou la police qui peut être différente.

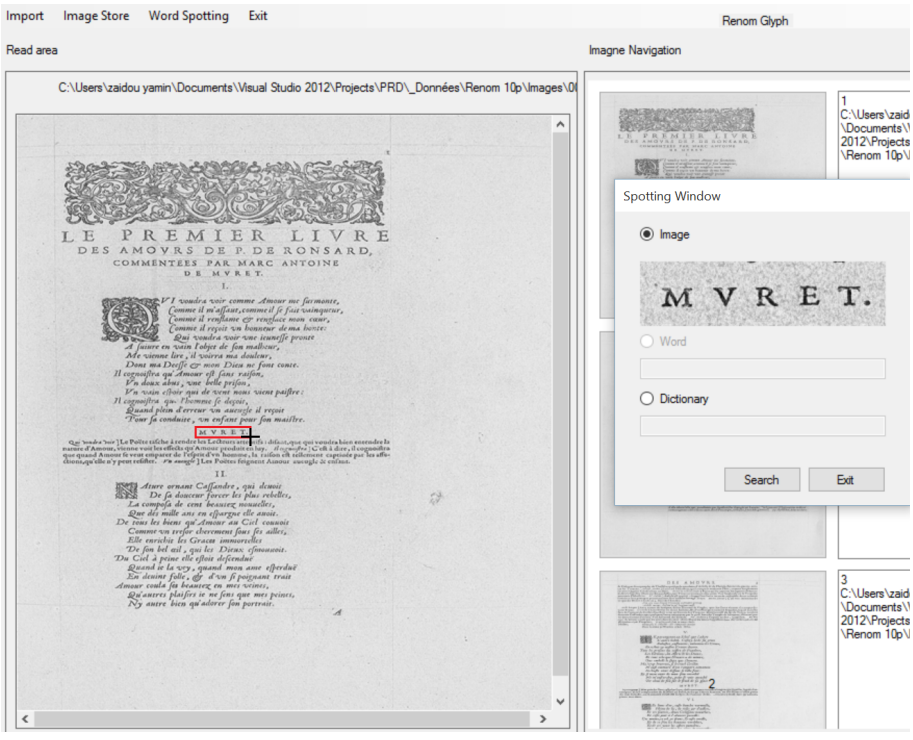


Figure 6 – *Fenêtres de recherche par sélection d'une zone*

Seules les fenêtres les plus pertinentes dans le cadre de ce projet sont décrites ici. Toutes les fenêtres sont détaillées dans le cahier de spécification [1] et dans le rapport de Loreen LAMBIN [2].

4.1.2 Outils de word-spotting

Les outils de word-spotting sont des programmes qui permettent d'effectuer la recherche d'un mot (représenté par une image) dans une masse de mots de même type. Ils effectuent ses recherches en trouvant les meilleures correspondances entre les caractéristiques de l'image et ceux de la base.

La version actuelle du logiciel implémente complètement l'utilisation de l'outil Renom développé par M. RAYAR, comme décrit dans le cahier de spécification de Loreen [1]. Cet outil permet d'effectuer une recherche par "mots-image" et par dictionnaire. Un deuxième outil développé par M. Bastien MEUNIER est partiellement utilisable par la plate-forme à cause du manque des fonctionnalités décrites en [7]

4.1.3 Configuration et exécution

Pour fonctionner, l'application utilise plusieurs références décrites dans le manuel d'utilisation. Elle possède 2 fichiers de configuration :

— ListOutil.xml :

Ce fichier contient les noms des différents outils de word-spotting, leurs emplacements et les types de requête possible que prend en compte chaque outil.

— BaseOutils.xml :

Il contient les noms des différentes bases d'images prises en compte par la plate-forme, leurs emplacements, et les outils pouvant être utilisés pour effectuer des recherches.

Au moment d'une recherche et avant l'appel d'un outil de spotting, un dossier "requête" est créé contenant l'image à rechercher et un fichier spécifiant son emplacement.

À la fin de la recherche, dans le dossier de la base d'image, un fichier resultat.xml est créé dans lequel sont décrites toutes les correspondances trouvées entre l'image de recherche et ceux de la base, les coordonnées en x et y, la largeur et la hauteur du mot.

4.2 Modélisation UML

À la fin du PFE de Loren et pour l'amélioration de la plate-forme, Une analyse UML a été faite pour re-architecturer les classes en modèle MVC et simplifier l'utilisation. Ces analyses présentées en annexe du rapport de Loreen [2] sont repris, vérifier et corriger en [20] pour les corrections et les améliorations de la plate-forme.

4.3 Dysfonctionnement et fonctionnalités manquantes

La dernière version récupérée présente des dysfonctionnements sur certaines fonctionnalités de la plate-forme. leurs corrections est prévue dans ce projet. De plus, de nouveaux besoins ont été établis en début de projet, afin de rendre l'application plus stable et indépendante des outils de word-spotting. Cependant dues à une contrainte de temps pour réaliser une vidéo de présentation et d'utilisation de la plate-forme, des corrections ont été apportées dans le code de l'application dans la première partie de ce projet.

4.3.1 Dysfonctionnement

Des erreurs d'exécutions apparaissent lors de l'utilisation de l'application :

- Un chargement trop long dans la fenêtre [Figure 3] : Comme elle propose deux modes de navigation,soit navigation par les noms des images ou une navigation avec les vignettes. le chargement du nom des images prenait beaucoup de temps car en parallèle, les images/vignettes était chargées dans l'autre onglet, la correction apportée était donc de ne pas paralléliser ces deux chargements car un utilisateur ne manipulera pas forcément les deux modes en même temps.
- Des erreurs de fuites mémoire : La première erreur aperçue lors de l'utilisation était une exception de fuite mémoire à cause des chargements d'images volumineuses(20 MB). A chaque chargement, il était nécessaire de réallouer de la mémoire. Effectivement la navigation d'une base se fait par l'affichage des images de celles ci. Une des corrections à apporter est de garder en mémoire l'image affichée, la précédente et la suivante.
- Des erreurs de liens : Lors d'une recherche et de l'appel à l'outil de word-spotting Renom, des exceptions de type emplacement introuvable étaient lancées. Ces erreurs étaient dues aux liens écrits en dur dans l'application et non générique. L'utilisation du debugger en mode à pas était nécessaire pour trouver et corriger ces erreurs.

Pour pouvoir présenter l'application par une vidéo de démonstration au mois de novembre 2015, certaines corrections ont donc été apportées. Elle ne sont pas toutes détaillées dans ce rapport et d'autres restes toujours à être corrigées et améliorées dans la partie développement du projet. Aussi dû à la reprise de plusieurs projets(l'application, les outils), des compilateurs différents et des versions différentes d'IDE étaient utilisés. Une configuration de ces dernières étaient nécessaire pour pouvoir poursuivre et maintenir plus facilement le projet.

Ces corrections m'ont cependant permis,lors de la lecture du code, de mieux comprendre l'application. Elles m'ont également permis de savoir reconnaître les différentes erreurs à corriger et de juger avec assez de précision le temps nécessaire à leurs corrections.

4.3.2 Fonctionnalités manquantes

Après une analyse et l'utilisation de l'application, des améliorations sont à apporter. Deux fonctionnalités importantes manquent et font partis des objectifs principaux du projet. Ces fonctionnalité sont :

- La Segmentation de ligne/mot : Cette fonctionnalité intervient à l'importation de la base, aujourd'hui absente l'application dépend fortement de base d'images pré-segmentées et traitées.
- Extraction de caractéristiques : L'objectif ici et de pouvoir extraire les caractéristiques des mots segmentés et ceux des requêtes (recherche) des utilisateurs. Ces caractéristiques permettront aux outils de word spotting de faire la correspondance.

Ces deux fonctionnalités ont fait l'objet d'une étude et sont présenté dans l'état de l'art de ce rapport[10] .

5 Les objectifs

L'objectif de l'application est de pouvoir disposer d'une plate-forme permettant, aux utilisateurs, la recherche d'un mot-image dans une collection d'images importée dans l'application. La plate-forme doit permettre l'utilisation de plusieurs outils de recherche tout en restant indépendant de ces derniers. Enfin une approche ergonomique sera adoptée afin de la rendre la plus simple et intuitive possible.

Pour cela il sera nécessaire de réaliser une adaptation de la conception actuelle pour permettre à l'application de pouvoir intégrer plusieurs outils. Ensuite il faudra apporter ces modifications à l'interface. Ces améliorations concernent principalement la complexité d'utilisation et le fonctionnement de l'interface elle-même. En effet, plusieurs procédés sont à revoir pour offrir à l'utilisateur une prise en main plus intuitive et une navigation au sein des bases d'images plus agréable et réactive. Enfin une restructuration selon le modèle MVC (Model View Controller) sera aussi nécessaire pour pouvoir disposer d'un code propre, facilement maintenable et améliorable.

En ce qui concerne l'architecture de l'application pour l'intégration de nouveaux outils, la version actuelle de la plate-forme intègre complètement l'utilisation de l'outil Renom, mais n'est pas adaptée pour l'outil CDP de B. MEUNIER. En effet cet outils nécessite d'avoir au préalable les caractéristiques des images.

L'importation des bases d'images est aussi limité puisqu'elle nécessite d'être segmentée et que les caractéristiques soient déjà extraites. Le plus gros du travail sera consacré à la segmentation et l'étude d'extraction des caractéristiques sur une image.

5.1 Partie Recherche

La première partie du projet est consacrée à la recherche et la rédaction d'un état de l'art sur les techniques ou méthodes qui seront utilisées dans ce projet. Elle permet la prise de connaissance du projet, l'étude des méthodes existantes, de leurs avantages et leurs inconvénients, ainsi qu'a la réflexion sur des possibles améliorations et adaptations de ces méthodes.

Ce document rapporte dans la partie recherche l'état de l'art :

- Sur les techniques de segmentation de ligne et de mot dans des documents manuscrits
- Les techniques d'extraction des caractéristiques dans une image
- Une étude de l'existant qui est présenté dans [la section 4]
- Une reprise/modification du modèle UML conçut par Loreen LAMBIN
- La méthode et gestion du projet

Les méthodes utilisées dans le contexte de notre projet seront plus détaillées. Des résultats sur des jeux de tests pour illustrer leurs performances, leurs complexités et leurs robustesses seront aussi présentées pour justifier les choix présentés en [19].

5.1.1 Objectif Bibliographique

L'état de l'art sera donc consacré sur les **méthodes de segmentation de ligne/mot** et d'**extraction des caractéristiques** sur des images représentants des documents anciens, pour utiliser/développer/implémenter les ces fonctionnalités dans la plate-forme.

Il s'agit d'un domaine de recherche du laboratoire Informatique de Polytech Tours, des documents/références repris donc de ses recherches ainsi que des bibliographes chercher et trouver sur les sites officiels de recherche sur internet sera cités en référence bibliographique pour plus de détails.

5.1.2 Analyse et conception

Un des objectifs dans la partie développement est de concevoir l'application en mode MVC, une refonte des interfaces et une analyse basée sur les documents UML de Loreen LAMBIN seront effectuées pour rendre la plate-forme indépendante des outils de word-spotting.

5.2 Partie Développement

La deuxième partie se portera donc sur les corrections, la reprise et le développement des différentes classes permettant de disposer d'une architecture MVC Le développement des fonctionnalités de segmentation et d'extraction de caractéristiques se fera dans un second temps.

5.2.1 Maintenance corrective

Même si certaines corrections ont déjà été apportées, un processus de remontée d'erreurs sera mise en place, afin qu'au cours du projet les utilisateurs puissent faire part des anomalies de la plate-forme et que les corrections correspondantes soient apportées rapidement.

5.2.2 Maintenance évolutive avec re-engineering

Les analyses et conceptions effectuées en première partie, permettront de pouvoir correctement améliorer l'application selon de bonnes normes de codage (comme de disposer d'une application développée avec MVC), les interfaces seront aussi simplifiées pour une utilisation simple, rapide et plus intuitives (par exemple, la suppression des nombreuses actions devant être réalisées pour effectuer une recherche).

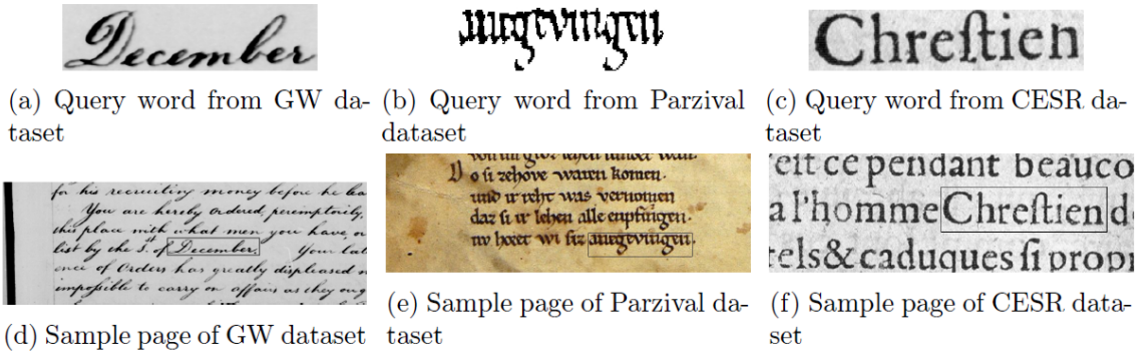
5.2.3 Maintenance évolutive : intégration de méthode/outil

Dans l'objectif de rendre la plate-forme complètement autonome des outils de word-spotting afin d'avoir la possibilité d'en ajouter/utiliser des nouveaux (qui devront respecter a minima les entrées/sorties dont l'application a besoin pour les utiliser). Une fonctionnalité de la plate-forme devra lui permettre, lors de l'importation d'une base, de pouvoir faire la segmentation et l'extraction des caractéristiques des instances segmentées. Ensuite lors de la sélection de la séquence de recherche, que ça soit par dictionnaire de mots ou par sélection de zone, elle doit être capable d'en extraire les caractéristiques. Un temps de développement assez conséquent sera nécessaire pour développer correctement ces deux fonctionnalités.

2 État de l'art

1 Principe du word-spotting

Un utilisateur doit être en mesure d'effectuer une recherche sur un mot, comme le montre l'image suivante reprise de la thèse [3]. Il s'agit d'exemples de résultats issus après une requête avec un outil de word-spotting.



(a) Query word from GW dataset (b) Query word from Parzival dataset (c) Query word from CESR dataset

(d) Sample page of GW dataset (e) Sample page of Parzival dataset (f) Sample page of CESR dataset

Figure 1 – *Exemple de résultat de recherche avec un outil de word spotting*

L'image suivante montre les différentes étapes pour effectuer une recherche.

En bleu les étapes d'initialisations d'une base d'images (ces tâches pourront être réalisées pendant l'importation d'une base dans l'application ou par un autre programme si la durée des tâches est trop longue) permettant de récupérer les critères de comparaison. En rouge c'est lors de l'émission d'une requête pour effectuer une recherche. La requête subit la même méthode d'extraction des caractéristiques. Puis la correspondance est faite entre ces caractéristiques et ceux de la base.

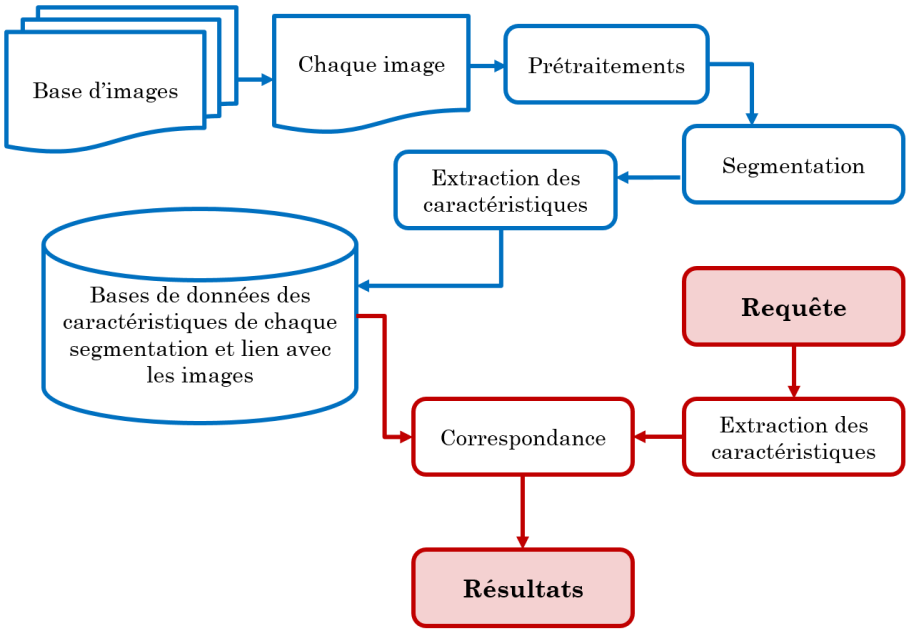


Figure 2 – *Principe du word-spotting*

Dans une base d'images chaque image subit des pré-traitements. Ceux-ci dépendent des méthodes de segmentation et d'extraction de caractéristiques. Ces pré-traitements peuvent aussi améliorer les résultats obtenus à l'issue de ces méthodes, cependant ils ne sont pas obligatoires pour certains algorithmes.

2 Segmentation en lignes/mots de documents anciens et manuscrits

L'objectif de la segmentation est d'extraire de l'image, des morceaux de lignes/mots ou caractères.

Cependant il existe plusieurs méthodes de segmentation des documents (journaux, revues scientifiques, magazines, lettre) ces méthodes sont citées dans la revue scientifique [WWW4] dont ce rapport s'inspire.

La segmentation dans une image reste difficile car comme le montre l'image suivante,ces documents présentent souvent des éléments perturbateurs.

Une forte proximité ou une grosse fragmentation va rendre la segmentation difficile car le problème sera de déterminer les distances inter lignes, inter mots ou inter caractères.

Un document avec des angles d'écritures différentes impactera beaucoup la difficulté à trouver le bon angle pour effectuer la segmentation.

Par rapport aux complexités de chaque document, des méthodes ont été développés pour pallier ces problèmes, chacune avec leurs avantages et inconvénients, certaines vont être présentées dans ce rapport.

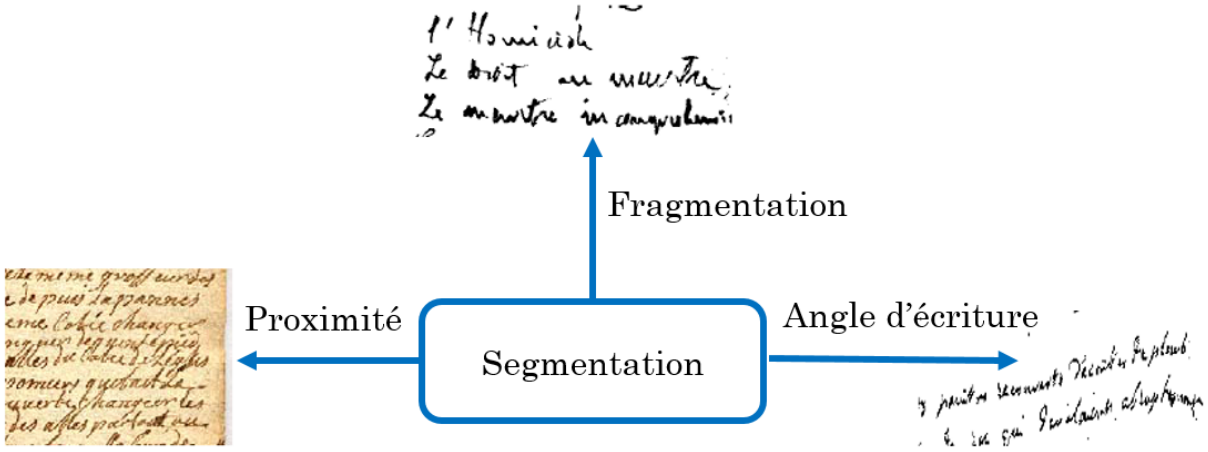


Figure 3 – *Axes principaux à prendre en compte lors d'une segmentation*

Voici une image d'un document d'une base (Renom) qui montre quelques difficultés qui peuvent être rencontrées :

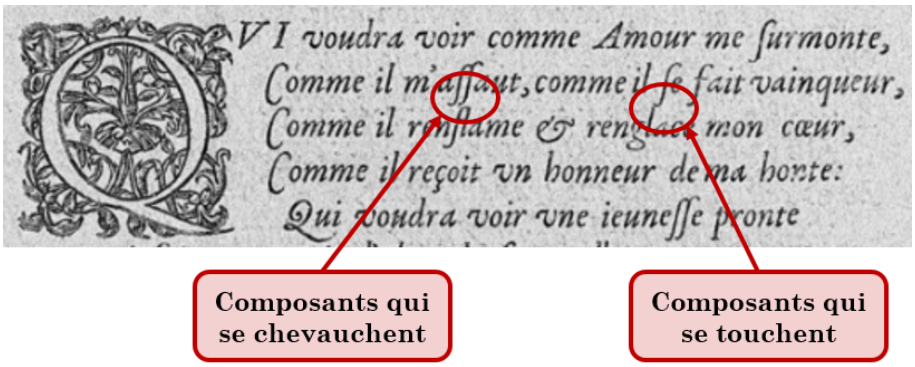


Figure 4 – *Interférences lors d'une segmentation des lignes*

Ces types d'interférences se retrouvent aussi dans certains exemples de segmentation des mots (lorsque deux mots se touchent). Elles peuvent être supprimées par des méthodes lors des prétraitement et facilité donc la segmentation.

2.1 Méthodes de segmentation

Comme le montre le schéma de la [Figure 2], la segmentation est appliquée dans chaque image pour fractionner cette dernière afin d'avoir que des lignes/mots ou caractères. Voici plusieurs méthodes de segmentation qui peuvent être appliquées dans les documents anciens et manuscrits.

2.1.1 Transformation de Hough

Appliquer en général pour détecter des lignes géométriques présentes dans une image, cette technique permet assez facilement de trouver l'angle d'écriture d'un document.

$$\rho = x_i \cos(\theta) + y_i \sin(\theta)$$

Grâce à cet formule, toutes les coordonnées des points d'une image sont parcouru pour être représenté dans un espace appelé l'espace de Hough ;

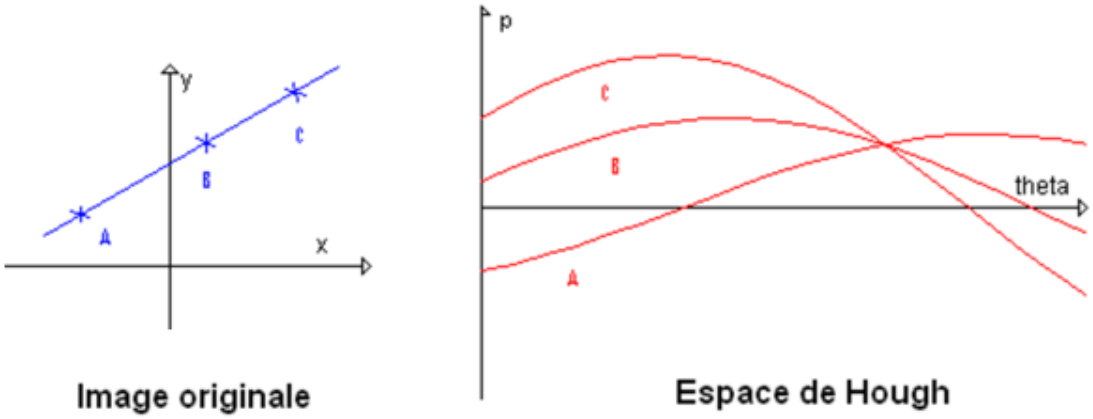


Figure 5 – *Exemple de transformée de Hough : source wikipedia*

dans cette espace chaque coordonnée (x_i, y_i) est donc représenté par la formule $\rho = x_i \cos(\theta) + y_i \sin(\theta)$ où :

- θ : l'angle d'inclinaison par rapport à l'axe des abscisses (x)
- ρ : l'éloignement de la droite par rapport à l'origine du système de coordonnées (x,y)

Chaque point de l'image est donc projeté dans l'espace, un point (x_i, y_i) est représenté par la sinusoïde qui correspond à toutes les droites auxquelles ce point peut appartenir. Dans un document manuscrit si les pixels noirs représentent des points sur un repère, avec la transformée de Hough on peut déterminer l'angle et les points/pixels appartenant globalement à une ligne.

Dans l'image ci-dessous, si l'on considère les points verts comme certains pixels d'une ligne, on observe qu'avec la transformée de Hough on obtient dans la zone bleue un croisement des courbes représentant les points et donc l'alignement de ces derniers.

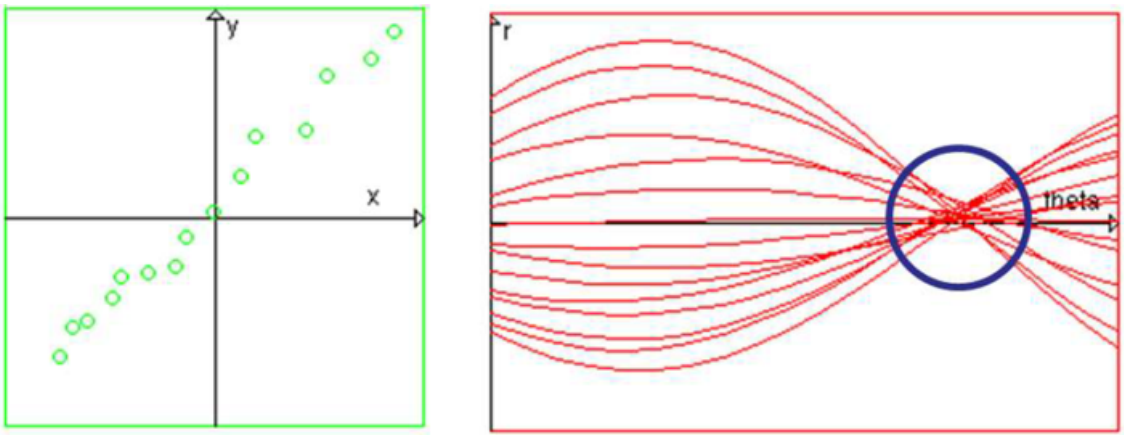


Figure 6 – *Exemple de transformée de Hough : source TSI*

Cette méthode est très efficace pour la détection des lignes et de leur l'orientation, cependant elle ne fonctionne pas ou n'est pas performante avec des lignes incurvées, des lignes avec des angles différents comme on peut le constater dans certains documents.

Likforman–sulem et al. en 1995 ont étudié une approche utilisant la transformée de Hough pour l'extraction des lignes des documents manuscrits anciens et utilise les composants au lieu des pixels. Aussi pour résoudre les problèmes de différence d'angle, comme expliqué dans la thèse [4]

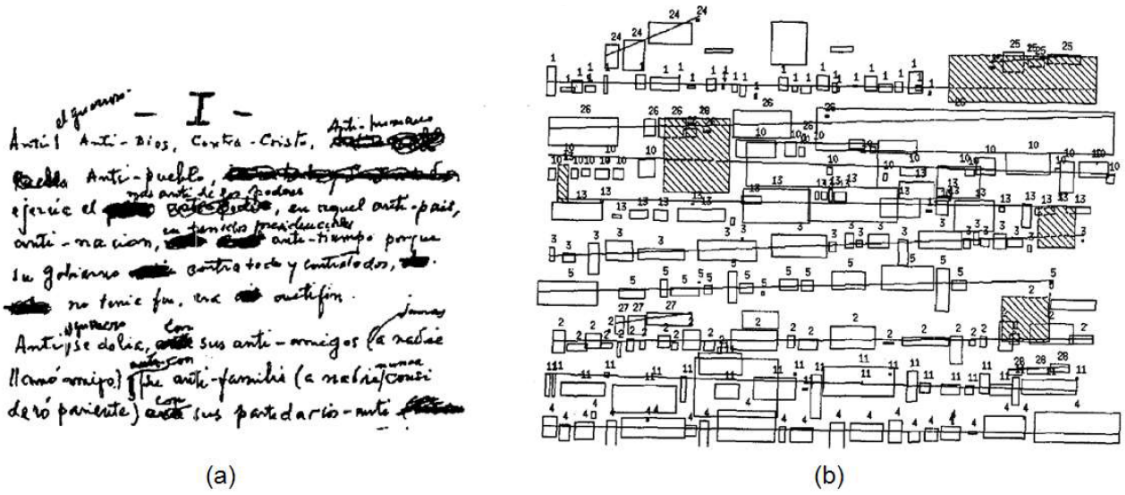


Figure 7 – *Extracation de lignes : (a) document originial et (b) résultat de la segmentation en lignes Likforman–sulem et al., 1995 source : thèse [4]*

2.1.2 Projection Profiles

Le principe de cette méthode est simple, il s'agit de compter horizontalement le nombre des pixels noirs dans chaque ligne de pixel.

$$Profile(y) = \sum_{x=1}^{n} f(x,y)$$

La méthode est souvent utilisée pour la segmentation des documents imprimés mais peut être appliquée dans notre contexte si les documents respectent certaines caractéristiques ou si ces derniers sont corrigés par les méthodes de prétraitements.

Voici un exemple visuel du résultat à l'issue de l'utilisation de la méthode (Image tirée de la webographie [WWW4]).

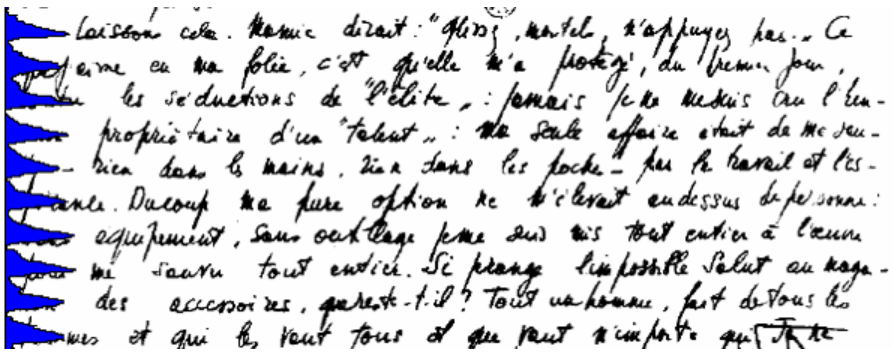


Figure 8 – *Exemple de résultat de segmentation de ligne avec la méthode de projection de profile*

On peut voir en bleu les pics des pixels noirs indiquant un nombre élevé des pixels noirs sur la ligne et des zones vides signifiant des fortes probabilités d'espace entre les lignes manuscrites.

En appliquant le même principe verticalement sur lignes extraites, on peut extraire les mots d'une ligne.

Les lignes avec des angles différents où des documents inclinés ne pourront pas être directement utilisés. Aussi des petits espaces entre les lignes, des lignes courtes ou étroites ne donneront pas forcément des bons résultats.

Il existe des prétraitements pour éviter ces défauts des documents :

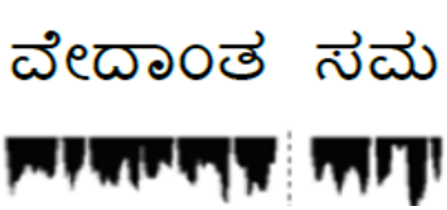


Figure 9 – *Exemple de résultat de segmentation de mot avec la méthode de projection de profile*

— Appliquer un filtre Gaussien ou Médian :

Pour éviter les pics trop élevés et les différences trop importantes entre les lignes contenant de nombreux pixels et celles contenant moins, on peut appliquer à la courbe obtenue après la projection pour lisser et obtenir une moyenne.

— Trouver le bon angle :

On peut appliquer la transformée de Hough pour trouver le bon angle, ou appliquer la méthode de projection sur des angles différents pour trouver le bon angle.

2.1.3 RLSA/Smearing

Cette méthode Run Length Smoothing ou Smearing Algorithm consiste à parcourir horizontalement un document rempli en noir les espaces de blancs qui se trouvent entre les pixels noirs se situant côte à côte. la technique permet rapidement de surligner les lignes d'une image. pour cela une variable de distance de pixel est défini pour noircir les petits espaces.

La figure ci-dessous montre une utilisation de la méthode de Smearing.

Figure 10 – *Exemple d'utilisation de la méthode de smearing*

Comme la détection est faite horizontalement, une correction d'angle est nécessaire pour l'utilisation de cette méthode cependant elle ne nécessite pas que l'image soit binarisée comme dans [Bourgeois, 1997, Bourgeois et al., 2001], ils ont proposé une méthode d'extraction des lignes qui prend aussi en compte une image dont l'angle d'inclinaison n'est pas forcément horizontal, les lignes sont extraites dans une image au niveau gris. Au lieu de connecter une série de pixels noirs et blancs, le gradient de l'image est dilaté selon la direction horizontale avec un angle d'inclinaison qui varie entre plus ou moins 30 degrés.

Donc pour cette méthode qui à montrer de très bons résultats et qui a été très étudiés par les chercheurs, son seul inconvénient et la variable de distance à paramétrer. Elle peut fausser l'extraction des lignes en donnant des gros blocs noirs si elle est mal paramétrée.

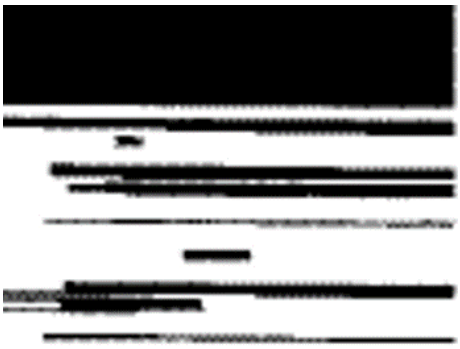


Figure 11 – *Exemple d'erreur de paramétrage de la méthode de smearing*

Cependant cela reste assez facile de la paramétrer. Une fois les lignes extraites on peut aussi appliquer le même principe pour extraire les mots.

2.1.4 Autres méthodes

— Classification par k-ppv :

Présenté dans la thèse [4] et appliqué dans les documents arabes. Deux études de [Zahour et al., 2004] et [Zahour et al., 2007] ont montré le fonctionnement de cette méthode. Dans la figure suivante, une première étape (figure a) et de diviser en colonne de même taille le document, ensuite pour chaque colonne (figure b), elles sont classées en trois catégories selon leurs caractéristiques, les blocs violets qui représentent les symboles diacritiques, les blocs rouge qui contiennent les corps du texte et les blocs bleus qui contiennent le chevauchement entre les mots des lignes voisines, enfin (figure c) les blocs sont regroupés entre eux en utilisant des distances euclidiennes, les blocs sont ensuite assemblé pour former des lignes.

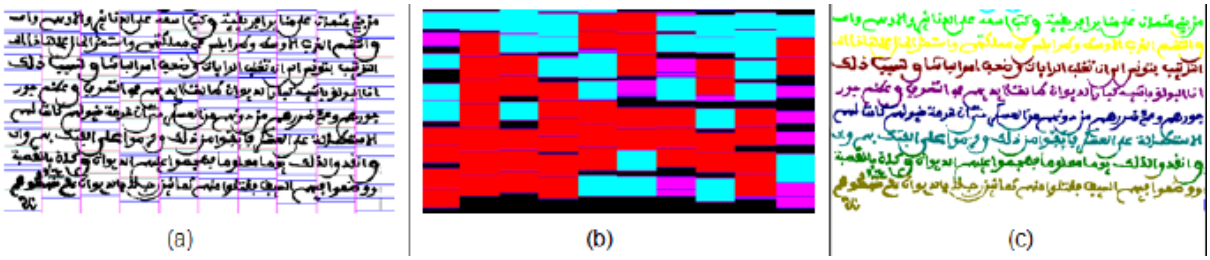


Figure 12 – *Étapes d'extraction de lignes : (a) maillage en plusieurs colonnes du document, (b) segmentation des colonnes en 3 types de blocs (petits en violet, moyens en rouge et grands en bleu), (c) résultat finale d'extraction de lignes [Zahour et al., 2004, Zahour et al., 2007] source : Thèse [4]*

Cependant pour cette méthode en sélectionnant les plus proches voisins, si pour un document, il y a une forte proximité, la méthode peut échouer.

3 Extraction des caractéristiques

L'extraction des caractéristiques est une partie primordiale pour que les outils de word-spotting puissent faire une bonne correspondance entres les instances et pouvoir avoir de bon résultats lors des recherches. Différentes études ont été mené dans ce domaine notamment à Polytech Tours avec la Thèse [3] de M. Tamnoy qui servira de source pour plusieurs informations dans la suite de ce documents.

3.1 Column Based Feature Extraction

Il s'agit d'une méthode très étudié dans la littérature et par plusieurs chercheurs dont [Marti et Bunke, 2001],[Rath et Manmatha, 2003], [Rath et Manmatha 2006], [Khurshid et al., 2012], [Frinken et al., 2012], [Rodríguez-Serrano et Perronnin, 2009], [Fischer et al., 2012]. Le principe est de décrire toutes les colonnes de pixel de chaque mot segmentés, cette description est faite à partir de 8 caractéristiques différentes extraite donc dans chaque colonne.

Voici un tableau récapitulant ces caractéristiques :

Table 1 – *caractéristiques extraites pour chaque colonnes. source [3]*

Sr. No	Feature set description
F1.	Somme des niveaux gris
F2.	Nombre de Transitions non-encre vers encre
F3.	Profile supérieur de l'image
F4.	Profile inférieur de l'image
F5.	Distance entre F3 et F4
F6.	Nombre de pixels noirs
F7.	Position du centre de gravité

Parmi ces caractéristiques, les caractéristiques F1 F6 ont été utilisé plusieurs fois dans différentes littératures décrites dans la thèse [3], mais les caractéristiques F7 et F8 ont été proposé par M. Tanoy.

- F1. : Cette première caractéristique permet de calculer et vérifier si la colonne est sombre par rapport à la moyenne de l'image
- F2. : Ici le nombre de transitions de blanc au noir est récupérer, ce qui peut être très intéressant pour détecter les trous entre les lettres, le début et la fin d'une lettre sans trou.
- F3. : Cette caractéristique permet de détecter ou commence la lettre dans le sens vertical s'il y en a une.
- F4. : Pareil que la Troisième mais ici on cherche à détecter la fin de la lettre.
- F5. : Dans cette caractéristique on cherche à calculer la distance entre le pixel noir le plus haut et celui du plus bas.
- F6. : On cherche ici à compter le nombre de pixel noir dans la colonne.
- F7. : Cette septième détermine le centre de gravité des pixels noirs de chaque colonne.

3.2 Slit Style HOG Feature

Dans cette méthode présenté dans la thèse [3] et dont un travail assez conséquent a été réaliser avec le document scientifique de Kengo Terasawa et Yuzuru Tanaka [WWW3] Elle est presque similaire à la méthode précédente. Au lieu de détecter des points clés dans l'image et récupérer ses caractéristiques. La fenêtre de taille fixe se déplace horizontalement dans toute l'image pour calculer l'histogramme du gradient orienté (HOG).

Voici une image illustrant ce principe :

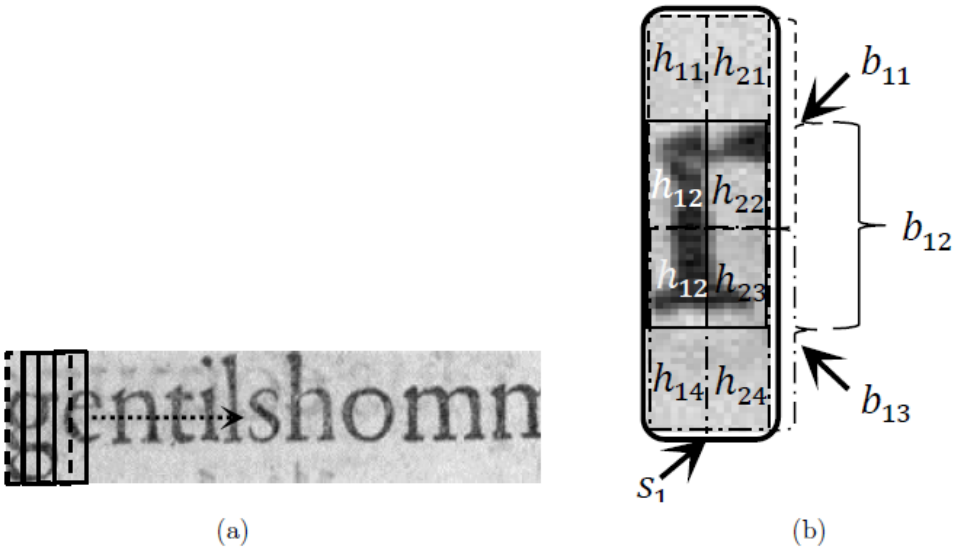


Figure 13 – *Extraction des caractéristiques sur un point clé*

3.3 Projection profile feature

Cette méthode plus appliqué dans l'extraction des caractéristiques sur des segmentation de caractères peut être adapter à notre cas. Le principe est de récupérer dans chaque caractère des histogrammes sur tous les profiles :

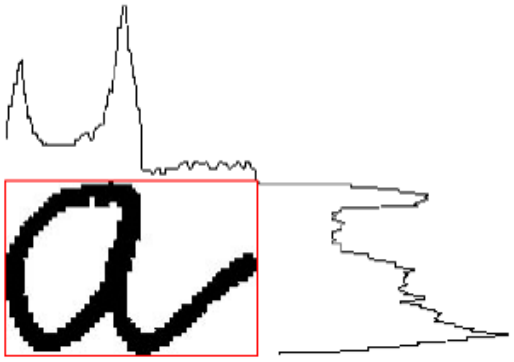


Figure 14 – *Projection Histogramme sur un caractère*

Dans la [Figure 14], il s'agit tous simplement de compter le nombre de pixel noir dans la colonne horizontalement et verticalement. Dans la [Figure 15], on compte le nombre de pixel de la bordure jusqu'au premier pixel rencontré et cela appliquer sur les 4 bordures haut, bas gauche et droite.

Figure 15 – *Porfile du nombre de pixel en la bordure et le premier pixel*

4 Choix et Propositions

Les choix fait dans cette partie dépendent du contexte du projet, des études réalisées dans la littérature, pour vérifier si les idées souvent re-utiliser et donc performantes.

4.1 Segmentation en ligne/mot

Comme le montre le tableau Table 2, dans notre contexte nous nous intéressons que des segmentation de ligne et mot sans apprentissage. Dans la littérature les méthodes les plus étudiées sont présentées dans le tableau suivant avec quelques informations récupérer à partir de mes recherches :

Table 2 – *Tableau de comparaison des methodes de segmentation*

	Difficulté à paramétrer	Complexité	Robustesse	Prétraitement
Projection Profile	Facile	Forte	Moyen	Oui
RLSA	Moyen	simple	Moyen	Oui
Hough Transformation	Facile	Moyenne	Moyen	Non

4.2 Extraction de caracteristique

Cette partie de la recherche aurai nécessité plus de temps de recherche mais par rapport aux recherches efffectés une utilisation de la méthode "Column Based Feature Extraction" pourra répondre au besoin du projet.

3 Analyse et Conception

Une partie de la partie recherche a été consacré à réfléchir sur l'évolution de l'application tant au niveau développement qu'interface. L'objectif est de faciliter l'utilisation de la plate-forme pour la recherche soit intuitif. Une amélioration du code est à réaliser pour évoluer sa maintenabilité, sa stabilité.

Voici une maquette de la nouvelle application, des possibilités et améliorations qu'elle apportera :

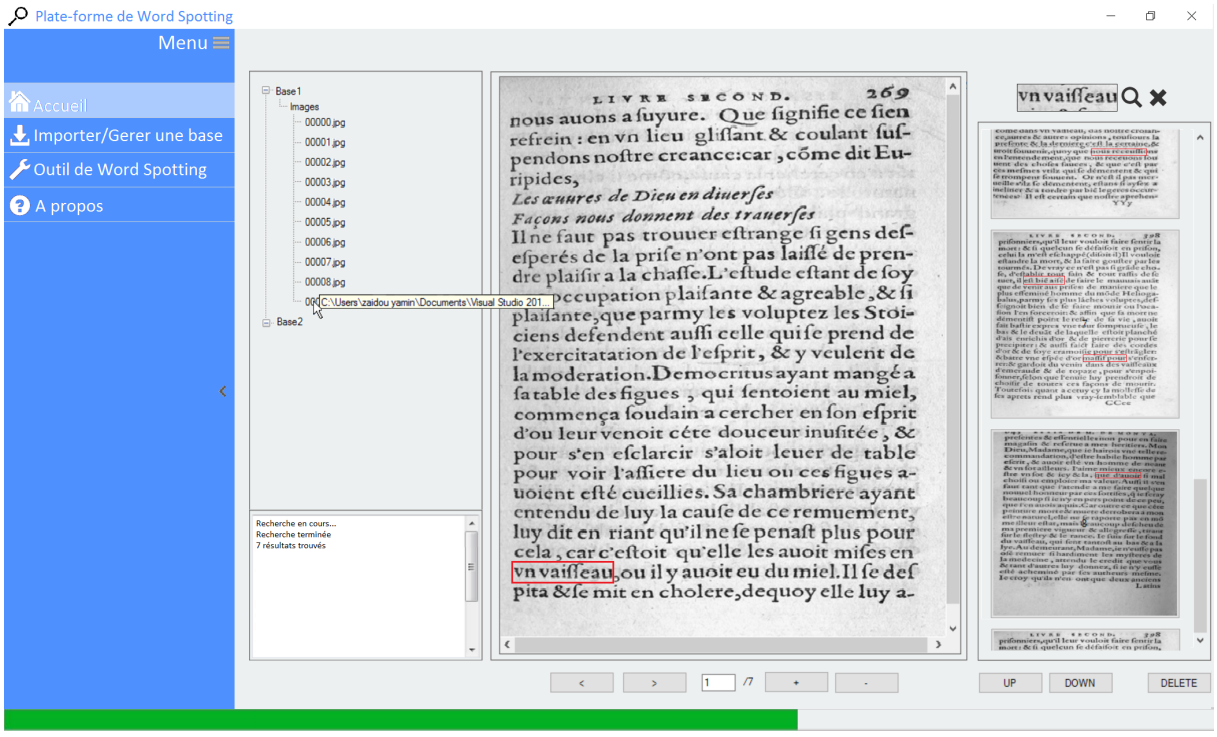


Figure 1 – *Maquette de l'interface d'accueil*

En utilisant le nouveau thème "MODERN UI" de Microsoft qui a été étudié pour améliorer l'expérience utilisateur et moderniser les applications, on trouve dans la maquette suivante une plate-forme avec les informations essentielles avec une barre Menu à droite rétractable, l'utilisateur peut facilement naviguer entre les différentes fenêtres de l'application, une barre de progression située en bas montrera le traitement en cours si la plate-forme effectue un travail.

Les différences avec l'ancienne version :

— navigation dans les bases d'images :

Au lieu de changer de fenêtre pour naviguer dans les bases, dans la nouvelle interface, l'utilisateur pourra voir l'arborescence et facilement changer de base directement à partir de la fenêtre principale.

— la fonction recherche :

Une image pourra permettre de visualiser l'image rechercher, l'objectif ici est encore une fois d'éviter les différentes fenêtres affichées les unes sur les autres et qui rendent difficile l'utilisation de l'application.

— le log :

Redimensionnable il pourra afficher plus de détails sur l'exécution de l'application.

La simplification des différentes informations présentaient à l'utilisateur et l'amélioration de la manière dont ils sont affichés pourra améliorer l'utilisation de l'application.

Dans la maquette suivante il s'agit d'un exemple pour montrer la suppression des différentes fenêtres qui étaient présentées à l'utilisateur, une seule fenêtre permettra dé d'afficher tous les informations dont l'utilisateur a besoin, tout en évitant de l'encombrer.

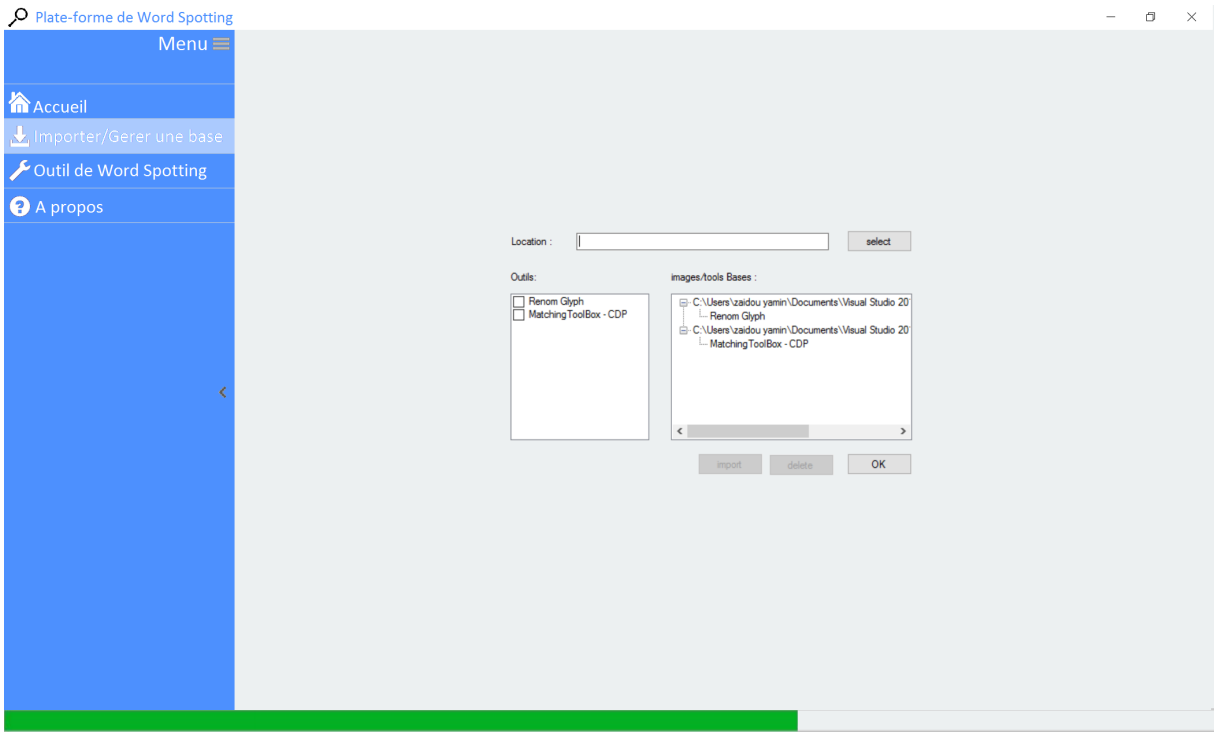


Figure 2 – *Maquette de la page d'accueil*

L'avantage de ces améliorations qui sont visible côté utilisateur n'impactera pas le code c'est-à-dire la suppression et la réunification de toutes les interfaces n'aura pas pour effet de créer un seul fichier avec tout le code car C# fournit des classes "UserControl" qui permettent d'ajouter simplement des morceaux d'interfaces dans une fenêtre. On aura une classe "UserControl" pour chaque ancienne fenêtre et une seule interface pour les gérer tous ; Dans cette unique interface (principale), on pourra donc naviguer entre les interfaces plus facilement.

Voici un diagramme montrant l'architecture des différents "UserControl" et la fenêtre ;

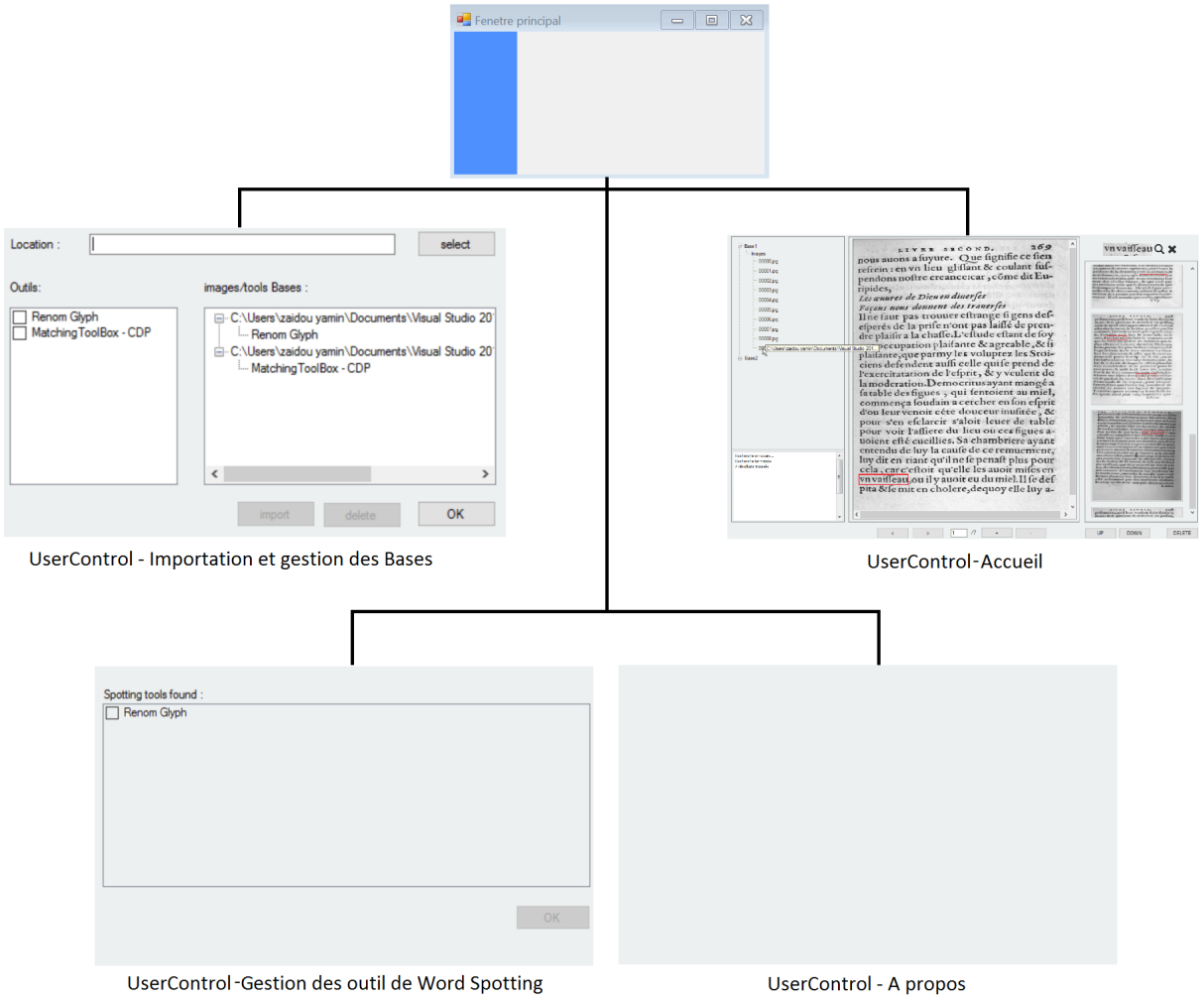


Figure 3 – *architecture des différents "UserControl" et la fenêtre*

Certaines fenêtres seront quand même affichées en plus de la principale, comme le clavier de recherche par dictionnaire et certains messages des utilisateurs.

le diagramme de classes que Loreen LAMBIN à proposer lors de son PFE est correcte et respecte le modèle MVC. les seules modifications apportées sur ce diagramme sont : la suppression de la classe "View Library", la modification des interfaces en "User Control", la création de la fenêtre principale la gestion de la base dans l'UserControlAccueil.

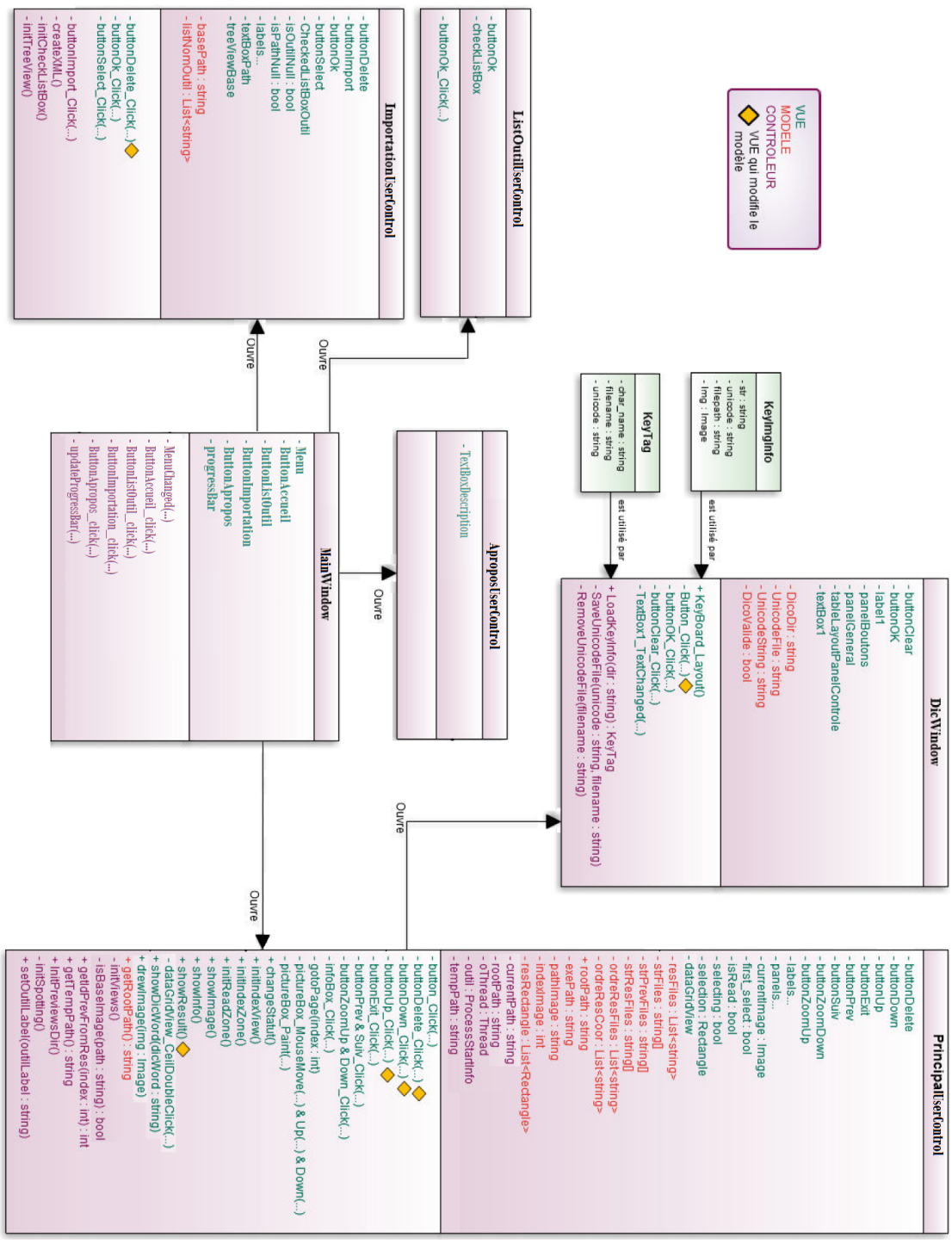


Figure 4 – *Diagramme de classes des interfaces existantes*

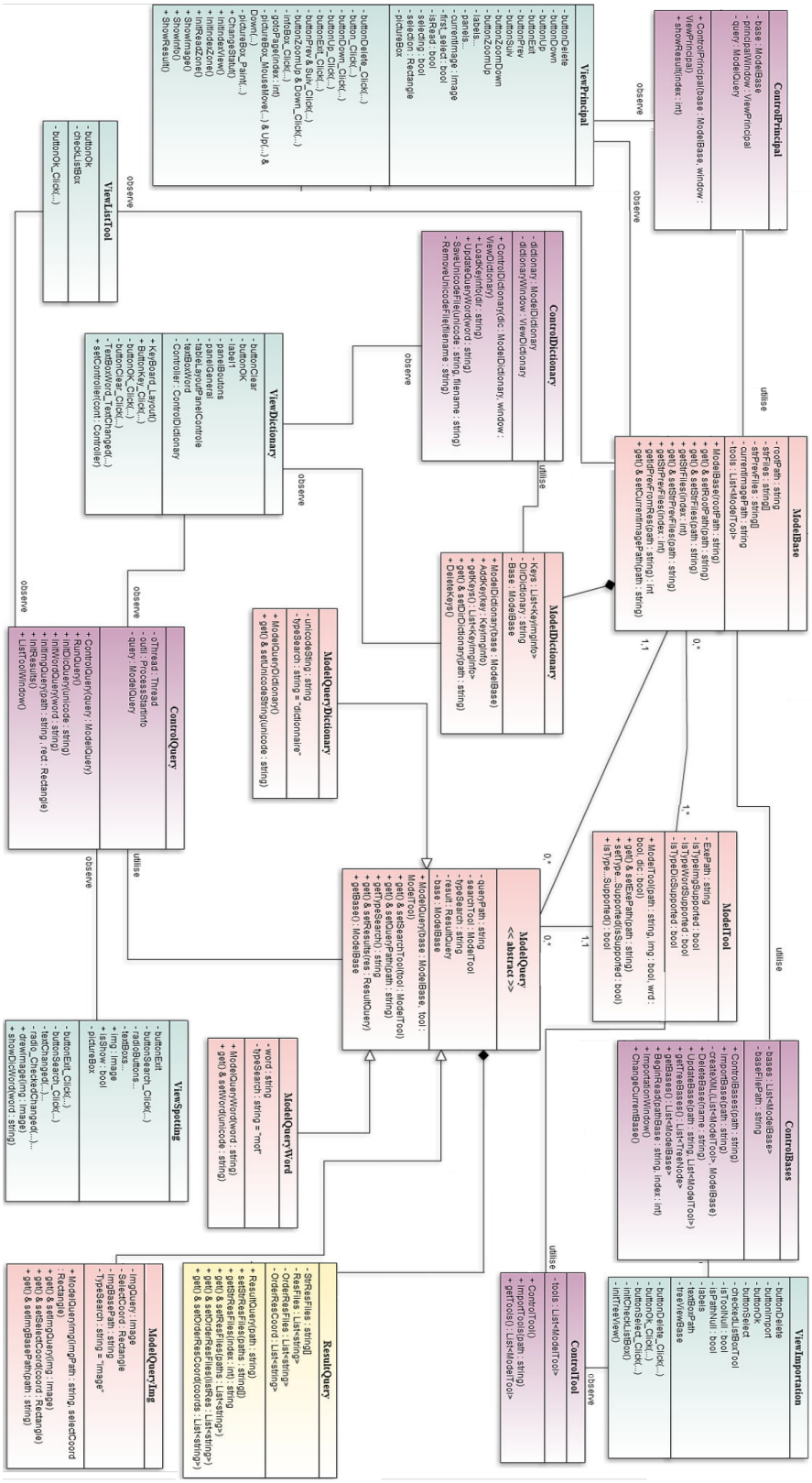


Figure 5 – *Diagramme de classes du nouveau modèle*

4 Bilan de la partie recherche

Cette première partie m'a permis de découvrir et d'avoir une idée sur le domaine de la recherche, la lecture des documents scientifiques et de pouvoir rapporter correctement ces informations. j'ai pu voir les difficultés de reprise de projet assez complexe. Les objectifs définis au début du projet ont été dans leur plus grande partie remplie, effectivement à cause de manque de temps je n'ai pas pu pousser mes recherches sur l'extraction de caractéristiques dans une image. cela est dû par les difficultés rencontrées lors de mes recherches.

Cependant les recherches effectuées, la prise en main, l'étude et l'analyse de l'existant ont été parfaitement exécutés et pose des bonnes bases pour la partie développement que ce soit dans l'intégration des nouveaux outils de word-spotting, la segmentation,l'extraction de caractéristiques ou dans l'amélioration des interfaces et de l'architecture du code.

De manière générale cette première partie du projet m'aura permis de découvrir l'ensemble des phases importantes d'analyse et de recherche d'un projet et de mettre en pratique les connaissances acquises. J'ai également eu la possibilité d'être force de proposition, concernant notamment la conception du nouveau modèle, ce qui m'a permis d'améliorer mon sens de la conception et de l'analyse. De plus, j'aurais appris à gérer un projet de cette envergure et à utiliser différents outils nécessaires à la gestion d'un projet.

5 Outils de gestion du projet

1 Versionning avec Visual SVN

La gestion de versions consiste à maintenir l'ensemble des versions d'un ou plusieurs fichiers. Pour la création de logiciels elle concerne la gestion des codes sources. Cette technique a été très utilisée pendant le déroulement de cette partie développement. Même si elle est surtout utilisée pour des équipes avec plusieurs développeurs et pour la fusion des codes. Son utilité dans ce projet a permis non seulement de sauvegarder les versions des sources dans un serveur mais surtout d'utiliser d'autres outils de gestion de projets comme Jenkins et sonar.

2 Intégration continue avec Jenkins

Jenkins est un outil au début créer pour le langage JAVA mais qui à très vite montré son efficacité lors des projets et a été très rapidement adapté pour plusieurs langages de programmation.

Jenkins a beaucoup évoluer depuis sa création et de nombreux plugins ont vu le jour pour augmenter son efficacité.

Son inconvénient est que son utilisation pour les projets Visual studio nécessite beaucoup de temps et de configuration du serveur surtout pour l'installation des plugins Microsoft (Build, Test...). Par contre une fois installé son plus gros avantage est l'automatisation des ensembles de pratiques utilisées en génie logiciel consistant à vérifier à chaque modification de Code source que le résultat des modifications ne produit pas de régression dans l'application développée.

Jenkins permet grâce à un plugin d'utiliser **SonarQube**[28] ; D'autres plugins existent pour ajouter des fonctionnalités mais n'ont pas été nécessaires pour ce projet.

3 Plugins Visual et Resharper

Visual studio 2012 est l'IDE utilisé lors de ce projet pour développer les différents programmes/logiciels, il s'agit surtout l'IDE que Microsoft fournit pour développer en c#,c++...

Il est très complet et permet d'ajouter plusieurs plugins afin d'aider et améliorer le code d'un projet, il existe aussi plusieurs plugins qui permettent d'éviter aux développeurs de sortir de l'ide pour effectuer des tâches du projet. Au sein même de visual studio on peut exécuter des **commandes**, afficher une **base de données**, ouvrir plusieurs types de fichier autre que ceux du projet comme un rapport,

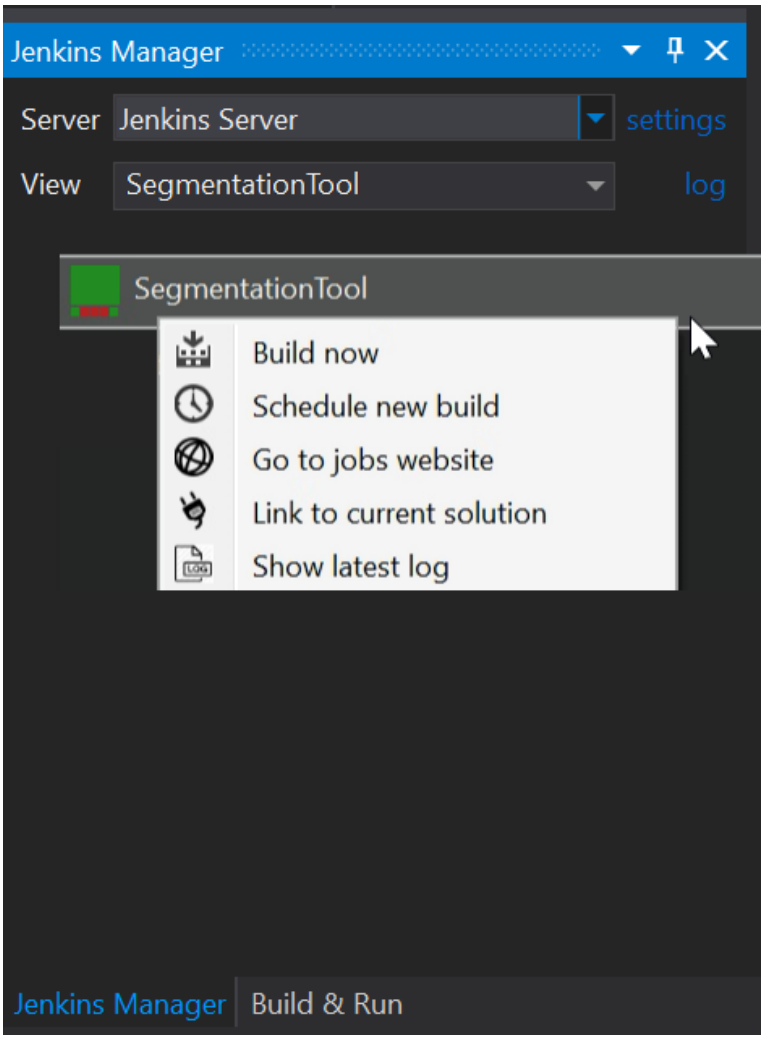


Figure 1 – *Plugins Jenkins pour visual Studio*

l'utilisation du gestionnaire de version **ViusalSVN**, l'affichage en écran scindé d'un fichier avec deux versions différentes...

Pour faire le lien avec Jenkins, il existe un plugins qui permet d'effectuer les tâches basiques de Jenkins :

Le projet à nécessiter l'utilisation de la bibliothèque **Aforge**, Visual studio permet grâce à **Nuget** (Un gestionnaire de package) d'ajouter directement au projet tout les Dlls nécessaires au fonctionnement de la bibliothèque **Aforge**. On peut aussi ajouter manuellement ces Dlls mais l'avantage avec Nuget est qu'il permet de vérifier si des nouvelles versions existe, de restaurer les Dlls manquant, un vrai gestionnaire de package.

3.0.1 Resharper

ReSharper est un plugin pour Visual Studio et est conçu par JetBrains. L'objectif principal de cet outil est d'augmenter la productivité des développeurs de logiciels. Il est utilisable sous une licence payante mais gratuit un an pour les étudiants, il est très utile et les améliorations faites par cet outil ne nécessitent pas sa présence lorsque la licence expire ou si un autre développeur reprend le projet. Voici un exemple d'amélioration de code proposer par Resharper :

```
1 count = 0;
2 foreach (Tuple<string, List<Rectangle>> result in _resultData)
3 {
4     if (result.Item1 == imagePath)
5         count += result.Item2.Count;
```

```
6 }
```

Le plugin détecte une amélioration possible sur ce code et propose

```
1 count = _resultData.Where(result => result.Item1 == imagePath).Sum(result => ↩
        result.Item2.Count);
```

l'inconvénient c'est que parfois la compréhension du code est difficile, surtout au début lorsqu'on ne connaît pas ou l'on ne maîtrise pas les expressions lambda ou les méthodes anonymes. L'avantage et qu'il y a vraiment moins d'indentation dans le code et moins de ligne de code, le développeur se retrouve plus facilement et à force d'avoir les propositions il finit par les comprendre et les utiliser d'où une des améliorations de la productivité. Cependant c'est par rapport à cet exemple mais dès qu'il y a une amélioration possible comme du code dupliquer une variable jamais utilisé, une zone de code mort, même le respect des conventions de codages, Resharper est capable de proposer une correction.

Il s'agit d'une des fonctionnalités de Resharper. Ce plugin propose aussi aux développeurs plus de détails sur des procédures ou des résultats comme les tests unitaires. Dans l'image précédente on peut voir

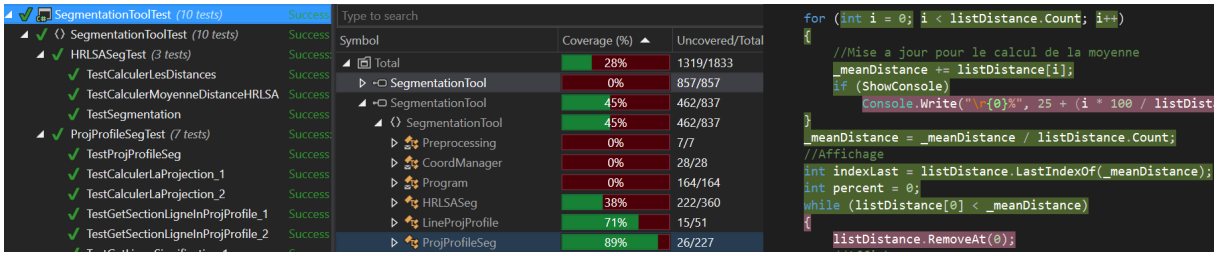


Figure 2 – *Plugins Resharper pour visual Studio*

les résultats des tests unitaires, le pourcentage de couverture de code par les tests unitaires et aussi directement dans le code :

Surligner en vert les lignes couvertes par les tests et en rouge les lignes non couvertes.

Ce rapport très précis permet aux développeurs d'améliorer leurs tests unitaires pour couvrir un maximum de code et ainsi avoir le moins de bugs possible lors de l'exécution. Il permet aussi aux clients/Encadrants d'avoir une garantie d'une application avec le moins de bugs possible et un respect des conventions de codage.

4 Qualité du code avec SonarQube

Sonar est un logiciel libre permettant de mesurer la qualité du code source en continu. Très utilise pour le projet car une fois mis en place permet d'inciter les développeurs à respecter les normes de codage.

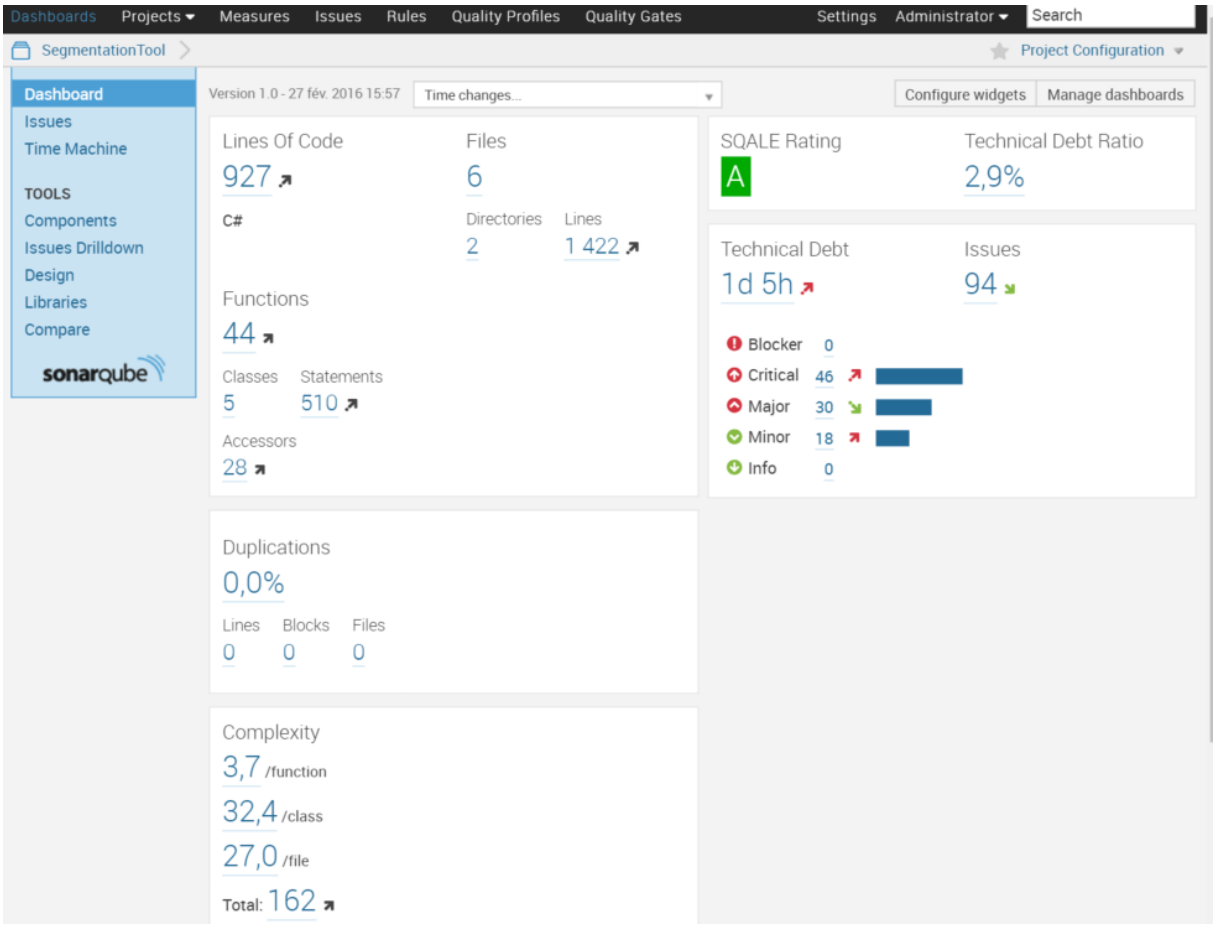


Figure 3 – *Rapport SonarQube*

Très rapidement après un commit d'un développeur par le gestionnaire de version et le build effectué par Jenkins, un rapport est présenté au développeur. On peut ainsi voir en détails quelle ligne de code créer une erreur et ainsi améliorer le code d'un programme.

6 Campagne de tests et résultats

Pour chacun des programmes développés dans ce projet, il a été question de respecter le plus possible les normes de codage, utiliser les outils de programmation pour avoir des applications stables. Cependant les tests sont les plus utiles,efficaces pour vérifier/corriger et assurer une application stable. Des campagnes de tests ont été mis en place et ont permis de corriger/améliorer le code fourni tout au long du projet.Ils ont aussi permis d'éviter la régression du code lors d'ajouts de fonctionnalités ou de mise à jour.

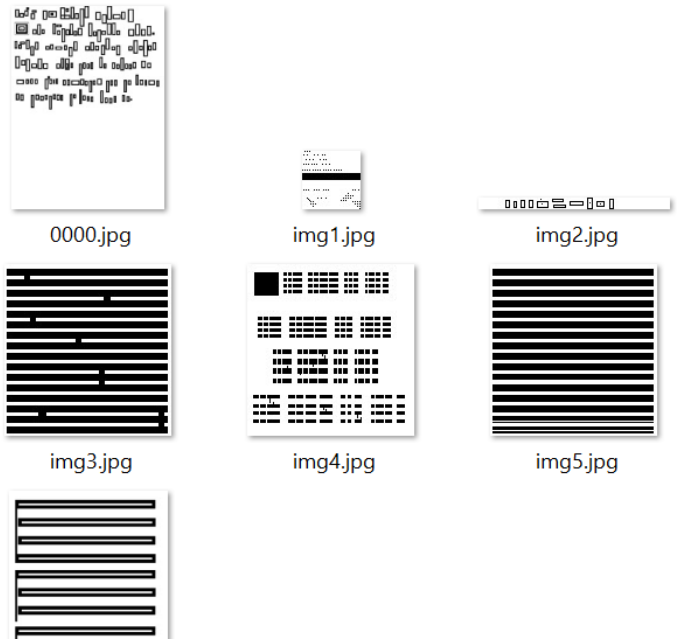


Figure 1 – *Base de test pour la segmentation*

Des images maîtrisées ont été créer pour valider les fonctions (ici pour la segmentation). Pour les images img3.jpg, img5.jpg et img6.jpg des tests sur les hauteurs moyennes des lignes, la distance moyenne entre les lignes, le nombre de lignes segmentées ont été effectué.

Pour l'image img2.jpg le test effectué est le calcul de la distance moyenne entre les composantes connexes, l'image prend en compte plusieurs paramètres comme deux composants qui se chevauchent.

Conscient qu'il faudrait une campagne de tests plus complète pour tester plus de fonctionnalités des applications développer dans ce projet, les tests les plus importants ont été implémentés et des corrections/améliorations ont été apportés dans les projets.

7 Mise en œuvre

Durant cette deuxième partie du projet,le développement s'est fait sur toute la chaîne, de la binarisation à la recherche avec un outil de word spotting. Dans ce chapitre sera expliquer les fonctionnalités développer/améliorer dans chaque programme, les résultats obtenus et les améliorations possibles pour le projet.

1 Binarization Tool

La binarisation fait partie des prétraitement avant la segmentation [Figure 2 (Chapitre 2)], elle permet de convertir une image en noir et blanc.

Dans ce contexte, une binarisation classique n'est pas la plus performante car elle se base sur un seuil pour effectuer la binarisation. En plus les résultats obtenus dépendent d'un seuil et l'image résultante contient parfois plein de défaut dû au mauvais choix du seuil.

Des algorithmes ont été proposé dans la littérature pour une binarisation spécifique à des documents contenant du texte.

Figure 1 – *Résultat de différents méthode de binarisation source : [WWW1]*

Un programme développé en c++ (au sein de l'école Polytech) reprenant la méthode de WOLF et al, a été repris/améliorer pour l'adapter au projet.

Les modifications effectuées sont :

— Programme compilable en c++ Visual studio :

Tous les programmes du projet étant développé sous visual studio, l'objectif était d'uniformiser. Comme le programme était développé en c++ Linux, diffèrent syntaxiquement du c++ de visual studio, l'objectif était de l'adapter et pouvoir le compiler avec la bibliothèque d'OpenCv.

— Modification du code pour allouer moins de mémoire Lors de l'éxécution :

  Des modifications ont été apporté dans le code comme des passages de paramètres par recopie changer en passage par référence. L'objectif utilisait le moins de mémoire possible.

— Affichage de l'état du programme :

  Un affichage en pourcentage de l'état du programme pour indiquer l'état d'avancement effectivement le temps d'exécution étant parfois très long avec plusieurs images a binarisé, cette information montre à l'utilisateur que le programme est toujours en état de fonctionnement.

— Adapter l'utilisation à un appel des autres programmes :

  le programme étant utilisé pour faire un prétraitement sur les images, il était nécessaire aux autres programmes de pouvoir l'utiliser.

Le programme prend en paramètre un dossier contenant des images, et binarise toutes ces dernières dans un sous-répertoire Binary.

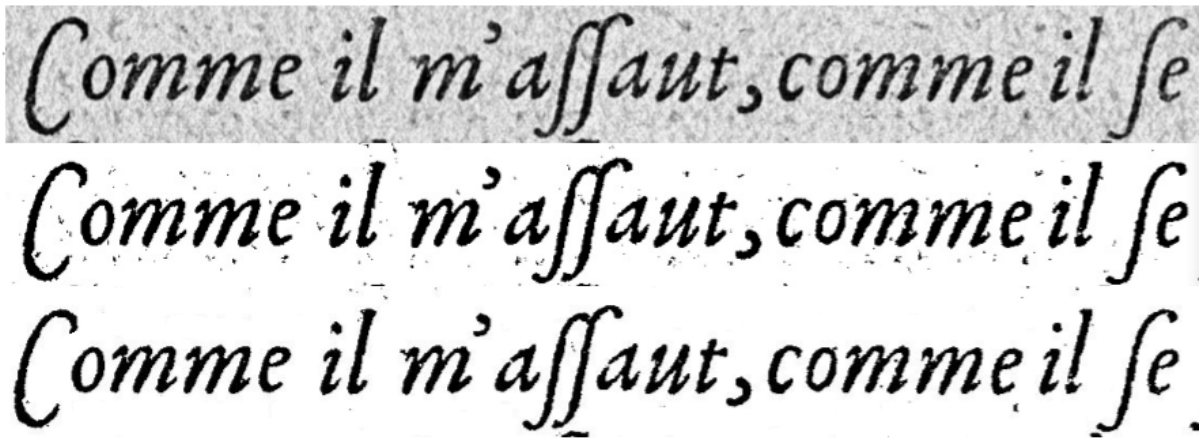


Figure 2 – *Résultat lors de la binarisation avec une méthode classique et l'utilisation de l'outil de binarisation*

Comme l'image le montre la deuxième ligne qui est binarisée avec une méthode classique et avec un seuil défini jugé être le plus adapté à l'image originale comporte quand même beaucoup de point noir car ils sont trop foncés dans l'image originale.

La dernière ligne est l'image issue de la binarisation avec l'outil. Très clairement la binarisation est beaucoup plus précise. Il s'agit d'une binarisation adaptée au contexte et qui permet surtout de faire ressortir le contenu textuel en éliminant le fond.

2 ExtracCaracs Tool

L'extraction intervient après la segmentation et lors de l'émission d'une requête [Figure 2 (Chapitre 2)], elle permet d'extraire donc les caractéristiques d'une image comme expliquer en [17]. Ces caractéristiques permettent de décrire l'image pour que les outils de word spotting puisse faire la correspondance entre deux images.

L'objectif est de pouvoir intégrer complètement dans la plate-forme l'outil word spotting (toolbox CDP de Bastien MEUNIER). La version précédente de la plate-forme était fortement dépendante des bases d'images dont la segmentation et l'extraction des caractéristiques étaient déjà faites pour utiliser l'outil CDP. Un extracteur de caractéristique a été développé par des projets précédents en c++. Cependant les résultats obtenus par ce dernier n'étaient pas exactes et le programme n'était pas encore adapté au format de l'outil CDP de Bastien MEUNIER.

Il a été question d'adapter le format des résultats pour l'outil de word spotting et par la même occasion, comme il était facile, le programme a été complètement restructurer et développer en c# tout en gardant la même logique.

La méthode d'extraction des caractéristiques utilisées est l'extraction à base de colonne [17]. Des améliorations ont été apporté dans les caractéristiques extraites, grâce à une réflexion avec M. Nicolas RAGOT et de la révue scientifique [WWW2]

Le tableau suivant reprends et décrits les caracteristiques extraites.

Table 1 – *caractéristiques extraites pour chaque colonnes.*

Sr. No	Description de la caractéristique	Normalisation	Type de l'image
F1.	Somme des niveaux gris	Oui	Niveau de gris
F2.	Nombre de Transitions non-encre vers encre	Non	Binaire
F3.	Profile supérieur de l'image	Oui	Binaire
F4.	Profile inférieur de l'image	Oui	Binaire
F5.	Distance entre F3 et F4	Oui	Binaire
F6.	Nombre de pixels noirs	Oui	Binaire
F7.	Position du centre de gravité	Oui	Binaire

En détails comment sont calculer les caractéristiques et normaliser :

A prendre en compte :

Un pixel Noir : valeur = 0

Un pixel Blanc : valeur = 255

Niveau de gris d'un pixel Noir : valeur = 0

Niveau de gris d'un pixel Blanc : valeur = 255

— F1. : La première caractéristique permet de décrire le niveau gris de la colonne, elle est calculée en sommant le niveau de gris des pixels de la colonne et ensuite est normalisée entre 0 et 1 sachant que le maximum est que tous les pixels sont à 255.

— F2. : Ici le nombre de transition de blanc au noir est récupéré, s'il n'y a pas de transition ou que le premier pixel est noir ensuite pas de transition alors la valeur est égal à 0.

— F3. : On cherche à retrouver le premier pixel noir par rapport au bord haut de l'image. Cette valeur est comprise entre 0 (premier pixel en partant du haut) et la hauteur -1 de l'image. elle est ensuite normalisée en divisant par la hauteur de l'image.
Une autre solution serait de récupérer la position par rapport à la boite englobante du caractère, comme le font dans l'article [WWW2]. L'avantage on évite le problème de découpage mais on perd la position/hauteur relative par rapport aux autres caractères.

— F4. : Pareil que la troisième mais ici on cherche à détecter la fin de la lettre.

— F5. : Dans cette caractéristique on cherche à calculer la distance entre le pixel noir le plus haut et celui du plus bas. Pour la normalisation cette valeur est ensuite divisée par la hauteur de la colonne.

— F6. : On compte le nombre de pixels noirs et divise par la hauteur de la colonne pour normaliser entre 0 et 1.
— F7. : Cette caractéristique doit renseigner le centre de gravité de la colonne, 0 s'il n'y en a pas. La position de tout les pixels noirs est sommée puis diviser par le nombre de pixel noir et enfin normaliser en divisant le total par la hauteur de l'image.

La dernière caractéristique présentées dans l'état de l'art [17] n'est pas utilisé mais devrait être testé pour vérifier si elle améliore les résultats de la recherche ou non.

Le programme est conçu pour être importé par les autres applications comme la segmentation et la plate-forme. Ainsi il s'agit d'une bibliothèque de fonction (générer en Dll) et importable facilement.

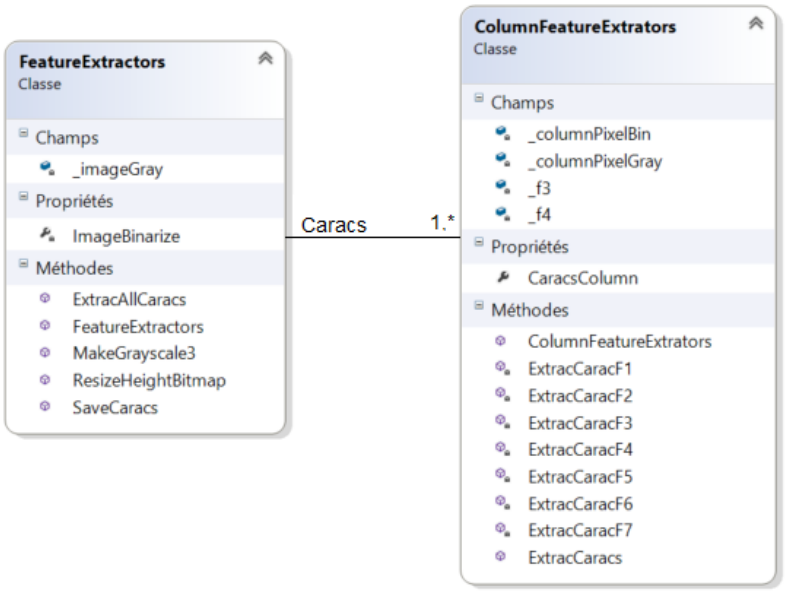


Figure 3 – *Diagramme de classe du programme d'extraction des caracteristiques*

La classe **FeatureExtractors** permet de gérer l'extraction des caractéristiques des images qui lui sont fournies. Il suffit de lui fournir lors de la construction de la classe l'image originale et l'image binarisée, elles sont ensuite dimensionnées à une hauteur de 70 pixels en gardant le ratio de l'image avec la méthode **ResizeHeightBitmap**. l'image originale est transformée en niveau de gris avec la méthode **MakeGrayscale3**.

Une fois la méthode **ExtracAllCaracs** appelé, toutes les colonnes sont parcourues et l'extraction des caractéristiques est ainsi faite surtout l'image. La classe contient donc une liste de la classe **ColumnFeatureExtrators** où toutes les caractéristiques sont stockées dans l'attribut **CaracsColumn**.

On peut accéder ainsi aux caractéristiques extraites ou les sauvegarder dans un fichier avec la fonction **SaveCaracs**, elle les sauvegarde en respectant le format de l'outil CDP de Bastien MEUNIER.

L'avantage est que s'il y a une modification/amélioration dans le code, tant que le diagramme de classe est respecté, il suffira de régénérer le dll et l'importer à nouveau dans les programmes l'utilisant. Il n'y aura pas de modification à faire dans ces programmes.

Les programmes l'utilisant (l'outil de segmentation et la plate-forme) auront ensuite les mêmes fonctions d'extractions de caractéristiques avec la même logique.

3 Segmentation Tool

L'objectif de la segmentation est de pouvoir séparer de l'image originale chaque ligne textuelle ou chaque mot/caractère. ainsi on pourra extraire les caractéristiques pour pouvoir les comparer avec celles de l'image requête.

La méthode utilisée va être décrite en dessous, il s'agit d'une combinaison de la projection profile [14] et du Horizontal RLSA [15] pour une segmentation en ligne.

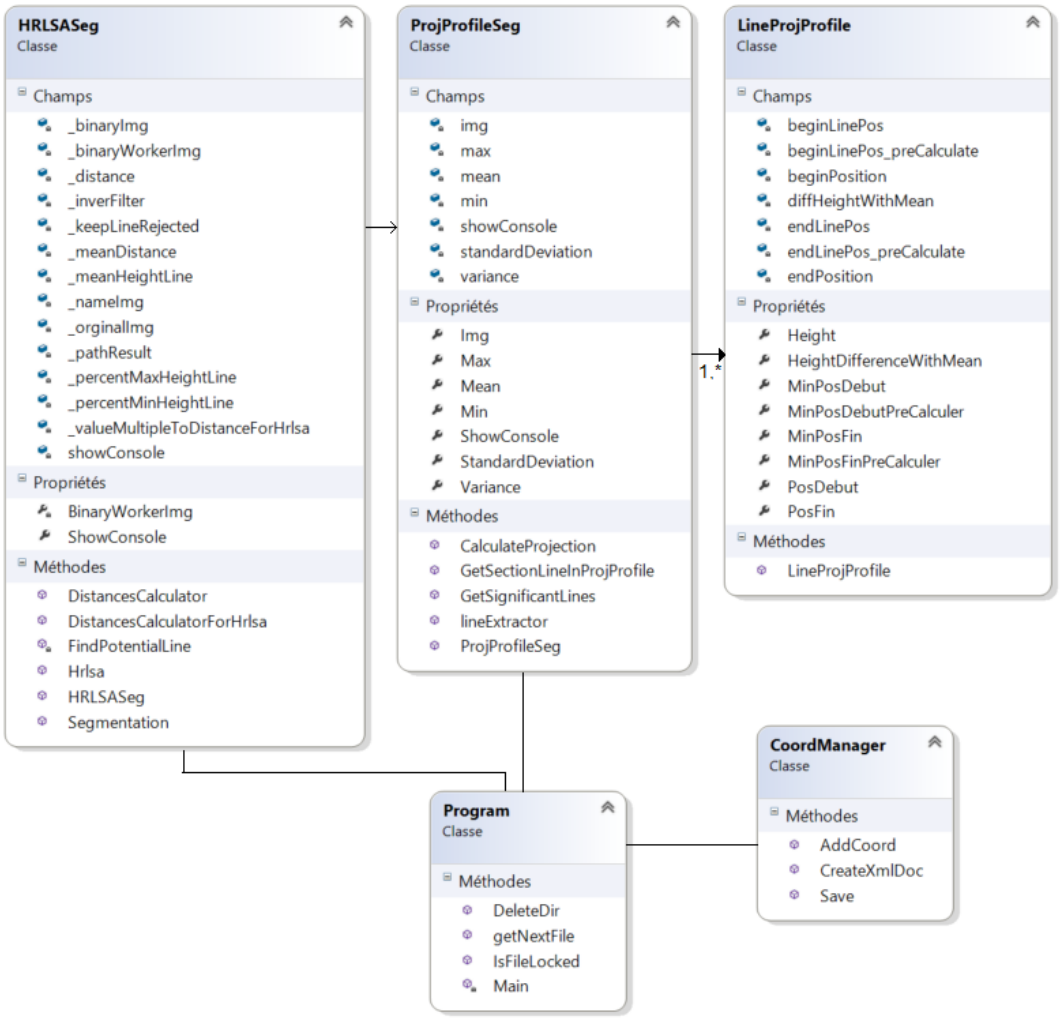


Figure 4 – *Diagramme de classe du programme de segmentation*

Le programme utilise la bibliothèque Aforge 2.2.5 (Version actuelle).

Pour pouvoir segmenter en ligne une binarisation de l'image est nécessaire, une fois l'image binarisée le programme récupère la projection profile Horizontale de l'image :

dans l'image suivante il s'agit d'une illustration pour comprendre le fonctionnement du programme. L'image est ici binarisé avec l'outil de binarisation et l'image originale n'est pas affichée.

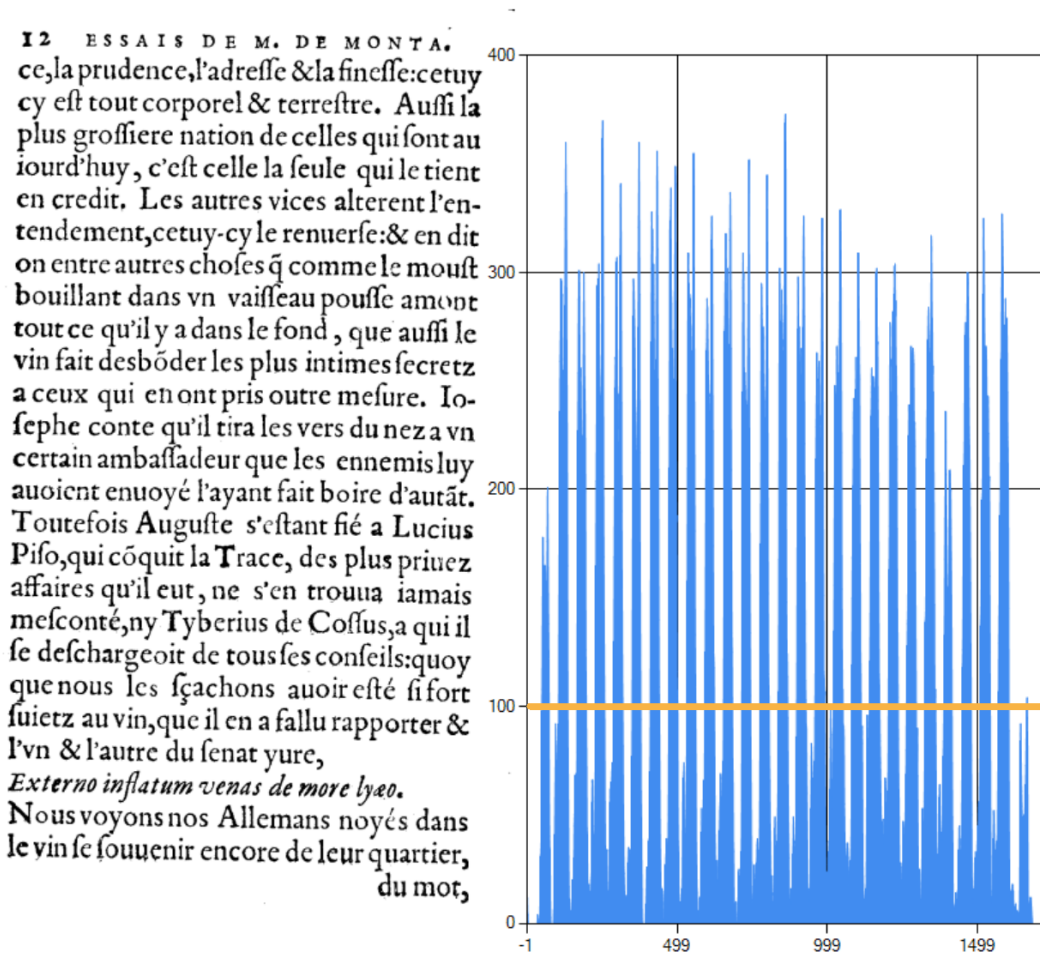


Figure 5 – *Projection profile d'une image*

Le programme grâce à la classe **ProjProfileSeg** arrive à calculer une première projection profile de l'image. il s'agit de compter horizontalement le nombre de pixels.

On obtient donc un tableau de valeur dont la taille est égale à la hauteur en pixel de l'image. Les valeurs représentent le nombre de pixels noirs dans la ligne.

La classe **ProjProfileSeg** fournit une fois la projection calculée, la moyenne (représentée dans l'image par la ligne orange), le minimum, maximum, variance et l'écart type de la projection.

Le programme récupère ensuite chaque bloc de pic bleu dépassant la moyenne (orange). Ces blocs sont représentés par la classe **LineProjProfile**. Un bloc bleu dépassant la moyenne est représenté donc par un objet de la classe **LineProjProfile** ou l'on trouve les positions verticales de début/fin du bloc. Ces blocs représentent souvent et très probablement des lignes dans une image. Une hauteur de ces lignes au niveau de la moyenne (ligne orange) est calculé.

Une fois toutes les lignes dépassant la moyenne récupérées, il y a une sélection des meilleures lignes (lignes significatives).

Pour cela les 30% des lignes dont la hauteur est plus petite sont supprimées. Une moyenne des hauteurs des lignes restantes est calculée.

Ensuite 30% des lignes dont leurs hauteurs sont trop éloignées de la moyenne des hauteurs sont supprimées.

On a ensuite 40% de lignes significatives dans le document.

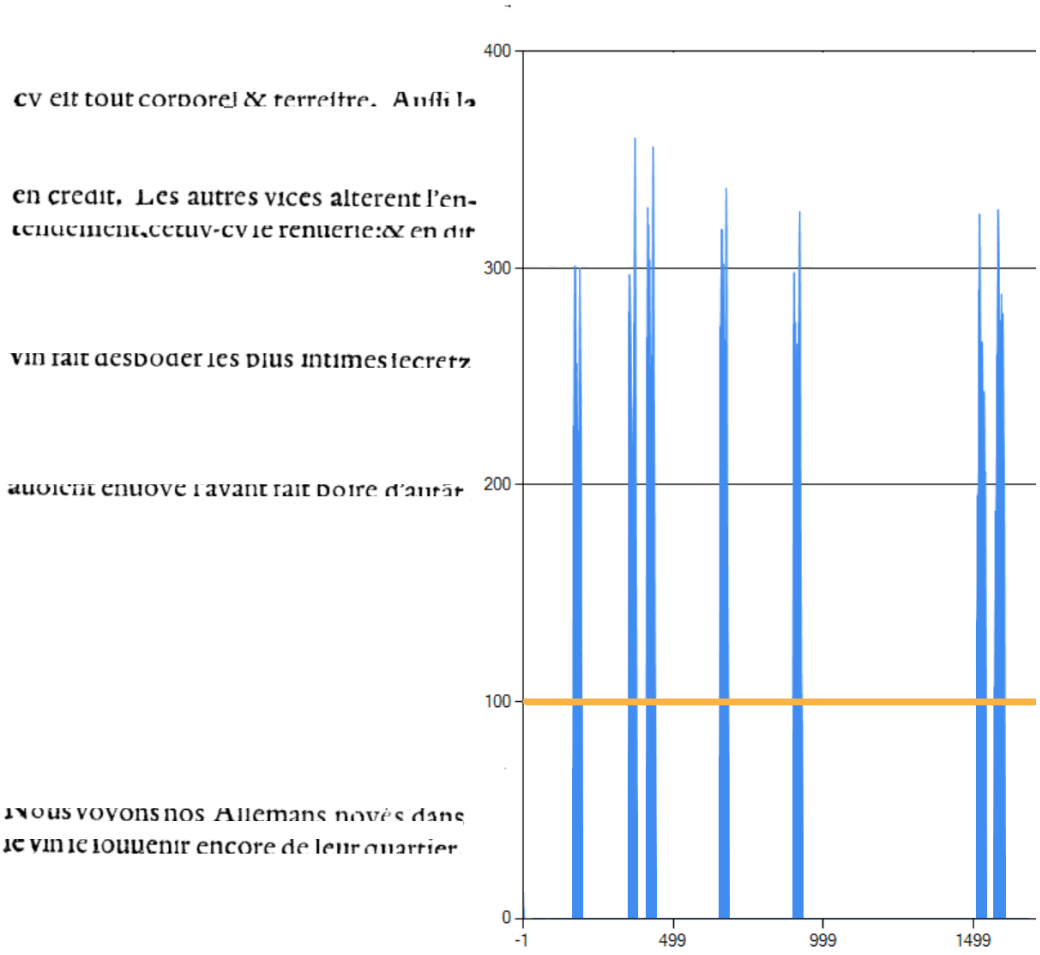


Figure 6 – *Résultat des lignes significatives*

Visuellement on voit que les positions début/fin récupérer coupent les lignes et on n'a pas la ligne complète en hauteur.

Cela ne pose pas de problème car la tache suivante est de calculer les distances entre les composantes connexes de ces lignes.

Une moyenne de ces distances est ensuite calculée avec au préalable 30% des plus petites et grandes distances supprimer.

Cette moyenne des distances est multipliée par un nombre choisi par l'utilisateur (par défaut 4).Le résultat représente la distance à utiliser dans la méthode de horizontale RLSA.

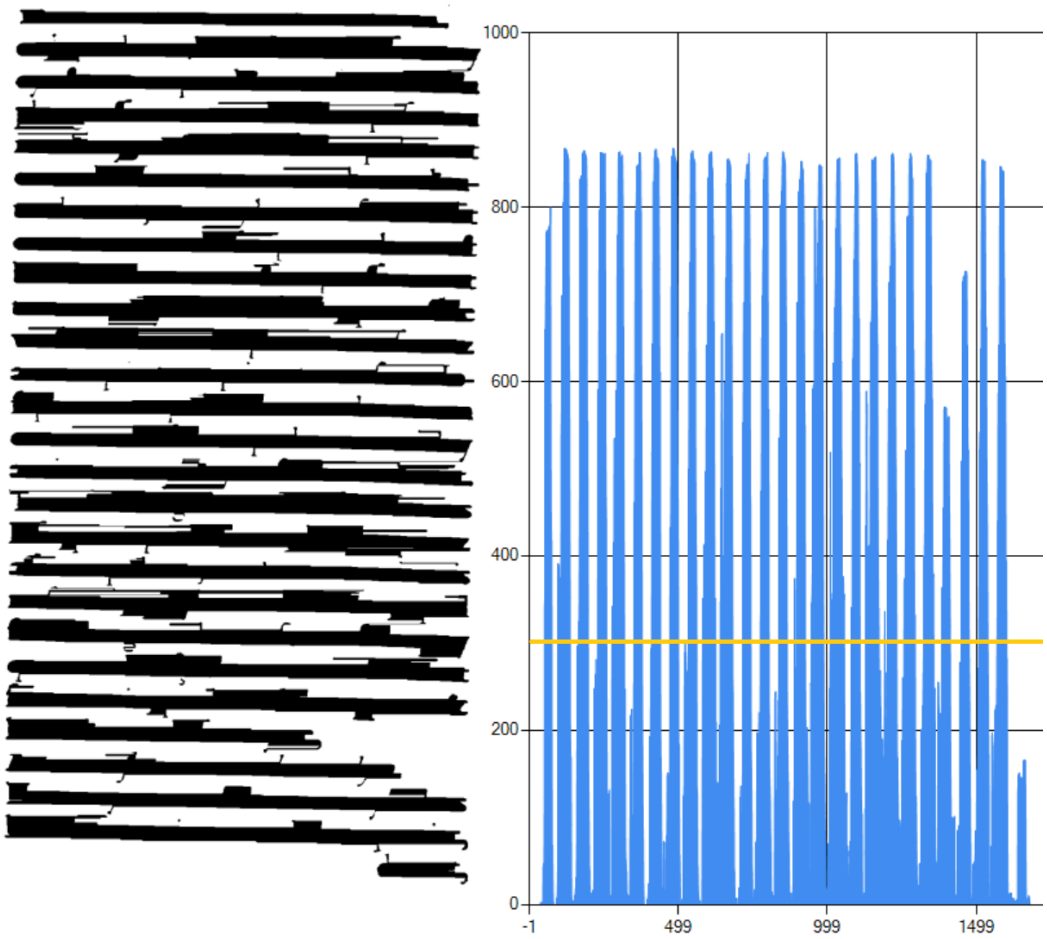


Figure 7 – *Résultat de l'utilistation du HRLSA*

Une fois le HRLSA appliqué dans l'image binaire de base. Le programme calcule les hauteurs des composantes connexes.

Une première moyenne des hauteurs est calculée, ensuite toutes les composantes connexes dont leurs hauteurs sont plus petites que la moyenne diviser par deux sont supprimées.

La moyenne des hauteurs est recalculée avec les composantes connexes restantes.

Ensuite pour chaque composante connexe dont sa hauteur est comprise entre l'intervalle :

Minimum = Moyenne des hauteurs - un pourcentage de la moyenne

Maximum = Moyenne des hauteurs + un pourcentage de la moyenne

les pourcentages sont donnés par l'utilisateur selon les images mais par défaut le pourcentage appliquer au calcul du minimum est 30% et pour le maximum à 40%.

Cette composante connexe est extraite de l'image originale car elle représente une ligne dans l'image.

Ceux dont leurs hauteurs sont inférieur à l'intervalle sont rejetés car elle sont jugées trop petite pour representer une ligne.

Et pour ceux dont leurs hauteurs sont plus grandes que l'intervalle. Le programme vérifie s'il n'y a pas des lignes potentielles dans ces composantes connexes.

Pour chaque composante connexe dont sa hauteur est plus grande que l'intervalle. Le programme applique la projection profile dans la composante.

On obtient donc :

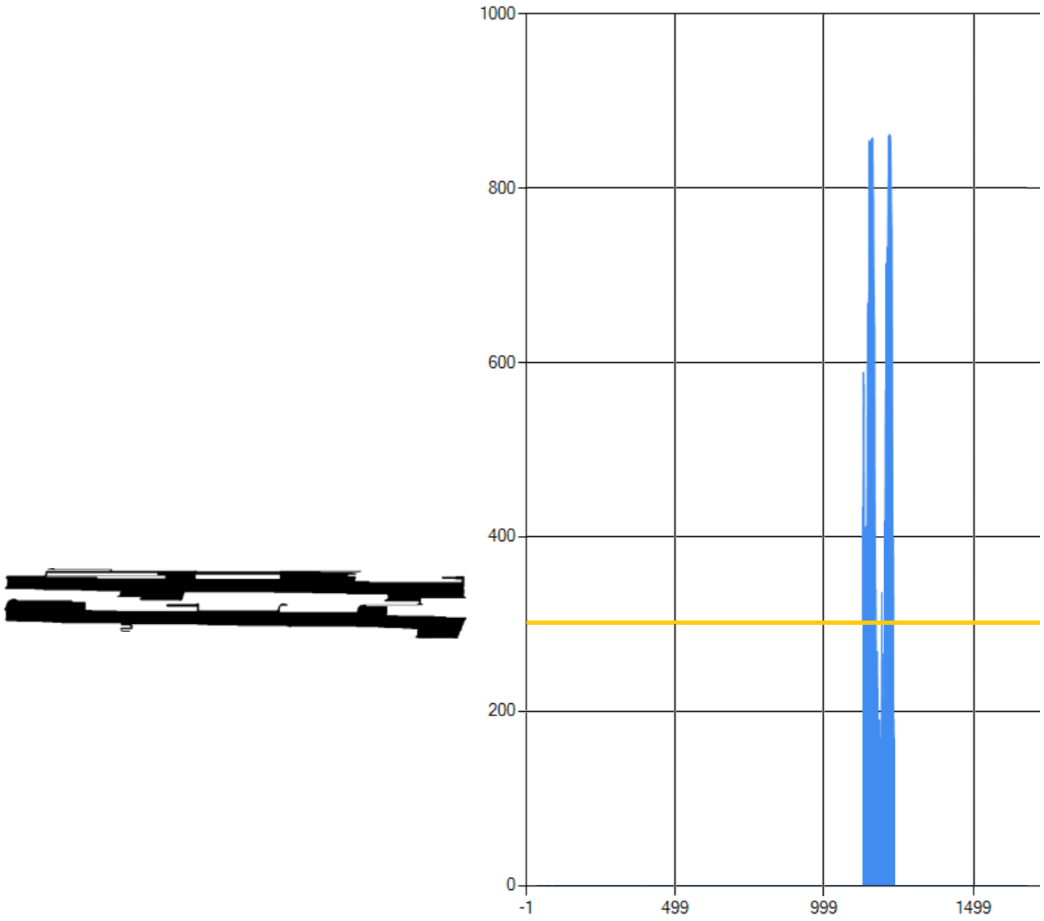


Figure 8 – *Projection profile d'une composante connexe trop grande*

Le programme vérifie la nouvelle hauteur de chaque ligne potentielle. La hauteur calculée est celle au niveau de la moyenne. Si une hauteur est comprise dans l'intervalle d'acceptation préciser au-dessus. Le programme accepte la ligne ou sinon la rejette.

Enfin comme vu précédemment les positions début et fin à ce stade ne sont pas parfaites par ce que le pic se trouve au milieu de la ligne et c'est pourquoi ces positions ne sont pas les plus parfaites.

Un des moyens de récupérer des positions début et fin de ligne plus précise est de procéder comme la méthode suivante.

Figure 9 – *Comment recuperer les positions debut/fin de ligne.*

À chaque ligne acceptée, on parcourt à sa droite comme à gauche le tableau de projection profile L'objectif est de récupérer la projection la plus petite autour de la ligne(en rouge dans l'image). Ces points sont plus précis sur le début et fin de la ligne.

Dans l'exemple suivant parmi les pics dépassant la moyenne il y en a deux qui n'appartient pas dans l'intervalle d'acceptation.

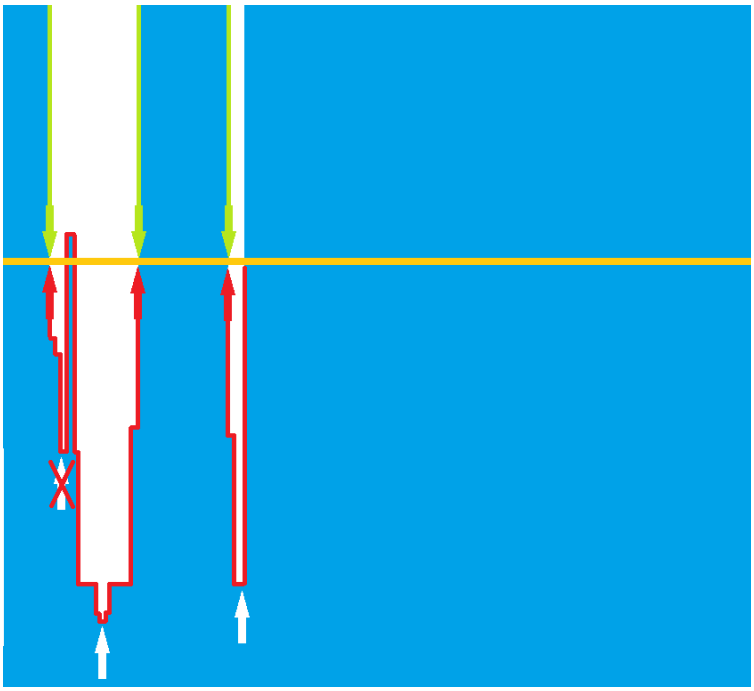


Figure 10 – *Projection profile contenant deux pics ne respectant pas l'intervalle d'acceptation*

Un pic dont sa hauteur est inférieure à l'intervalle est ignoré et les hauteurs des lignes autour sont recalculées. Et l'inverse n'est pas vrai car souvent il s'agit de figures dans les documents.

Cette méthode de segmentation est certes complexe et il existe beaucoup d'améliorations pour augmenter son efficacité et diminuer sa complexité, mais au cours des tests effectués elle a donné des résultats très positifs.

Pour la base **Vaisseau**, **100%** des lignes sont segmenter.

Pour la base **Renom**, plus de **95%** des lignes sont segmenter avec des figures ignorés.

Une fois toutes les lignes d'une image segmentée, le programme sauvegarde les images segmentées en respectant le format décrit en dessous. Les images de la base se trouvent dans la racine du projet.

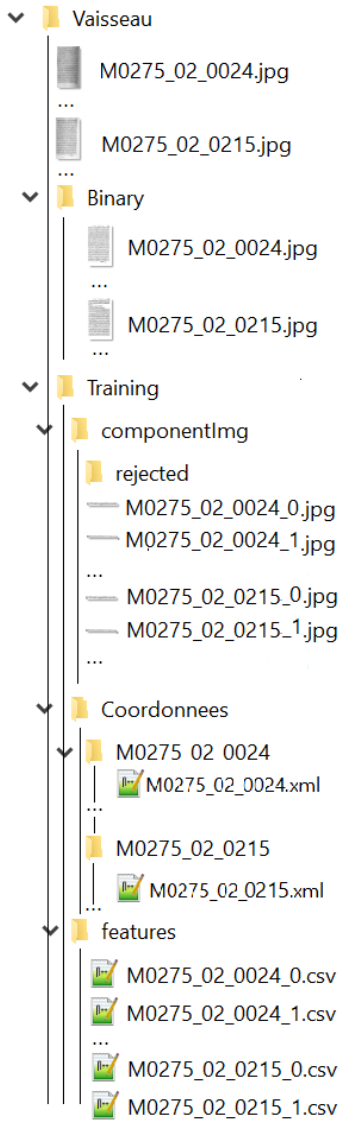


Figure 11 – *Dossier resultant de l'outil de segmentation.*

Les images binarisées dans le dossier binary.

Toutes les images segmentées sont dans le dossier training/componentImg, ceux rejetées dans Training/-componentImg/rejected. Le format des noms des images segmentées est : le nom de l'image suivi d'un underscore puis un index.

Dans le dossier Coordonnees/nom de l'image se trouve un fichier XML qui détaille la position x, y dans l'image de base de chaque composante connexe segmentée.

Enfin le dossier features contient les mêmes noms que les composantes segmentées mais il s'agit des fichiers XML contenant les caractéristiques extraites grâce à l'outil d'extraction.

4 Word spotting platform

La plate-forme à été complètement re-imaginer avec comme objectifs :

- Intégration des nouveaux outils développer.
- Re-factoriser le code en respectant le modèle MVC et le diagramme de classe
- Améliorer l'expérience utilisateur en simplifiant l'utilisation et en ayant un design plus moderne

4.1 Maintenance évolutive avec re-engineering

La maintenance évolutive devait permettre d'avoir une application stable, facilement maintenable et simple d'utilisation

4.1.1 Respect du modèle MVC

De base dans un projet visual studio c#, il n'y a pas vraiment de modèle MVC. Et ces projets ne sont pas vraiment adaptés pour respecter ce modèle. En fait un projet Windows forme respecte presque un modèle Vue/contrôleur simplement. Sur cette base le modèle est juste rajouté pour completer et avoir un Modèle MVC et pouvoir gérer les données lorsque les contrôleurs le demandent.

La partie Vue contrôleur reste inchangée et pour pouvoir gérer les données voici un exemple de classe dans le modèle.

```
1   class BasesManager
2   {
3       public BasesManager()
4
5       public List<Base> GetAllBase()
6
7       public Base GetBase(string rootPath)
8
9       public void AddBase(string rootPath, List<Tool> listTools)
10
11      public void DeleteBase(string path)
12
13      private static XmlDocument LoadOrCreateXmlDoc()
14  }
```

Ce modèle permet de gérer les données d'une base. L'avantage est que les contrôleurs n'ont pas à effectuer eux-mêmes les requêtes mais laisse les modèles s'en charger.

4.1.2 Séparation des taches dans des Users Controls

L'une des vrais modifications dans la version précédente sur la refactorisation est la délégation des taches dans des users controls. Un user control est en fait considérer comme une partie de fenêtre Windows forms. L'avantage est qu'elle est facilement intégrable dans les fenêtres parentes et permet d'alléger le code. La version précédente, presque toutes les taches était exécutée dans une même fenêtre, l'inconvénient c'est que dans le code, il y avait beaucoup de ligne de code et la compréhension du code, la maintenabilité était difficile. En utilisant les users controls on délègue des taches à des parties de fenêtre qui s'occupent eux-mêmes de les exécuter. On a une séparation claire et facilement compréhensible dans le code. Exemple pour la plate-forme la fenêtre mère s'occupe de la navigation entre les fenêtres filles :

- Home
- Import/Manage Base
- Tools Word Spotting
- About

Chaque fenêtre fille s'autogère. Dans la fenêtre Home, on peut aussi trouver une fenêtre fille :

qui gère l'affichage d'une image, la sélection d'une image requête, l'affichage des résultats et le zoom.

Très clairement on voit une plus grande aisance lors de la lecture du code, il y a effectivement largement moins de ligne de code par fichier et chaque grosse fonctionnalité est factoriser par rapport au reste du code.

4.1.3 Avantages du nouveau design

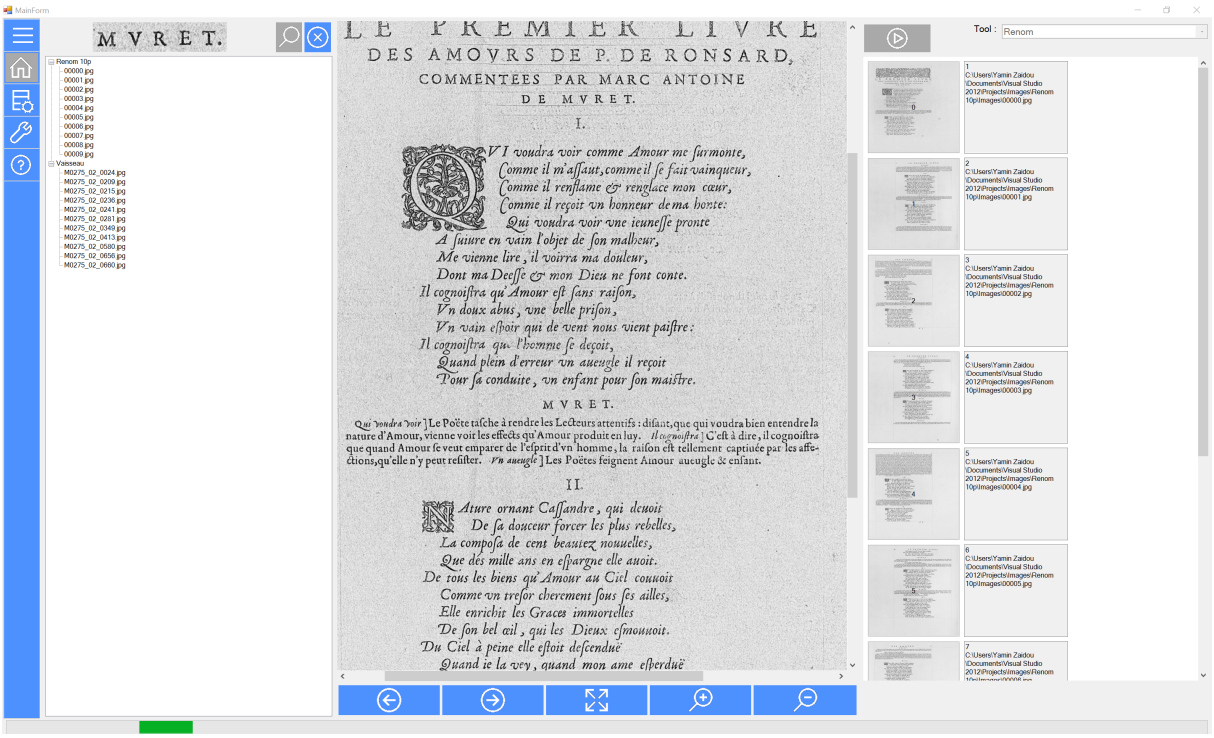


Figure 12 – *Fenêtre Home, Nouveau design*

Par rapport à l'ancienne version, lorsqu'il fallait effectuer une action, trop de changement d'interface et de clic était nécessaire, dans la nouvelle version, il était question de simplifier cette utilisation.

Tout est faisable dans la même fenêtre, elle possède un design plus sobre et plus facile à prendre en main.

De nombreux raccourcis intuitives ont été gérer pour faciliter l'utilisation.

4.2 Maintenance évolutive : intégration de méthode/outil

La nouvelle plate-forme a repris beaucoup des fonctionnalités de l'ancienne version, tout en les améliorants ou en les adaptant à la nouvelle version.

Comme l'ajout/suppression des bases ou des outils de word spotting ne se fait plus par la modification des fichiers XML mais directement dans l'application. La plate-forme est donc plus complète est intègre l'outil d'extraction des caractéristiques pour pouvoir utiliser l'outil CDP de Bastien MEUNIER.

4.2.1 Recherche avec les outils de word spotting

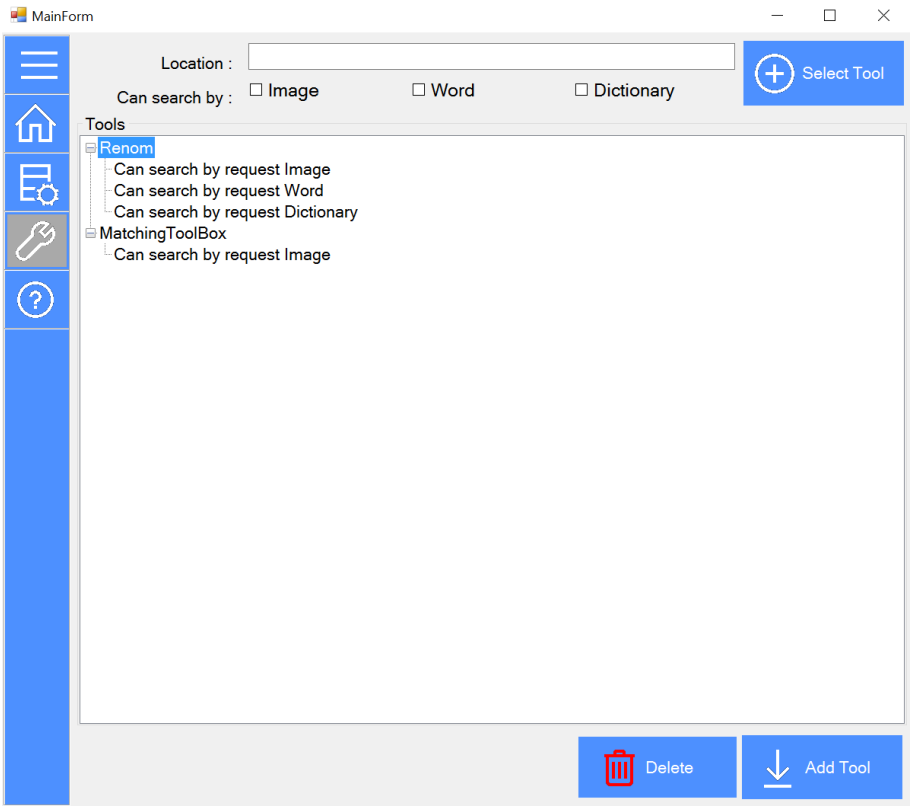


Figure 13 – *Fênetre de gestion des outils*

Les outils de word spotting peuvent être facilement intégrer dans la plate-forme tant que lors de l'utilisation ils respectent le format de requête et des résultats de la plate-forme.

La recherche se fait comme dans l'ancienne version avec création d'un dossier request. L'image recherchée et cette fois les caractéristiques de cette dernière sont présentes dans le dossier. L'appel aux outils de word spotting se fait toujours avec un appel de leur exécutable.

Le résultat est attendu dans la racine du dossier de la base d'image.

Les tâches principales de la plate-forme sont fonctionnelles, mais ne sont pas parfois complètes à cause du manque de temps pour la finaliser.

La gestion de la taille de l'image lors de la sélection d'une image requête ou lors de l'affichage des résultats n'est pas encore au point dû à la gestion dynamique du positionnement et de la taille de l'image. Cependant la sélection et l'affichage fonctionnent très bien lorsque l'image n'est pas en mode plein écran. Gérer pour l'instant une amélioration serait de permettre de sélectionner et d'afficher les résultats lorsque l'image est en plein écran.

4.2.2 Affichage des résultats

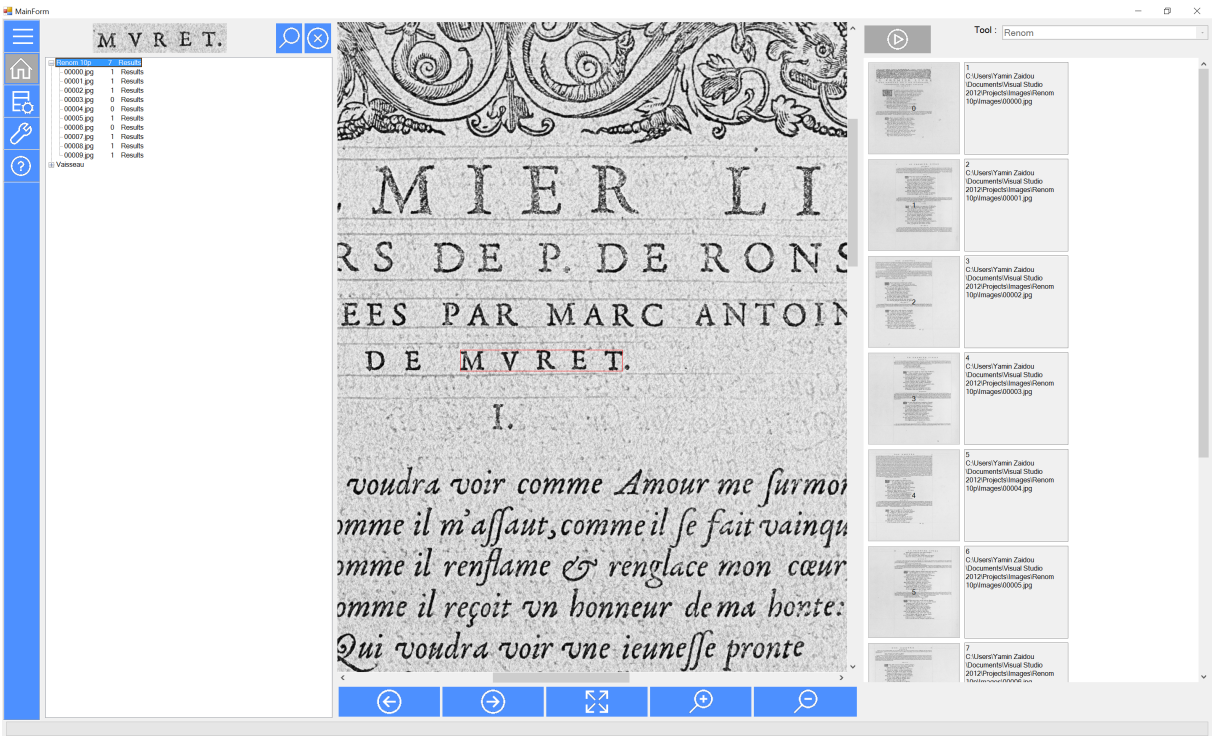


Figure 14 – *Fenêtre Home, Affichage des résultats*

L'affichage des résultats se fait comme la version précédente avec ici toujours l'image requête afficher pour pouvoir comparer facilement la requête et les résultats. Le nombre de résultats est affiché cette fois-ci dans le treeview à droite.

8 Gestion du projet

1 Méthode de suivi de projet

La méthode agile permet des livraisons incrémentales de fonctionnalité au travers de Sprints, avec cette technique décidée avec monsieur N. RAGOT, des taches pourront être réalisées testées et vérifiées directement pour une amélioration plus rapide et stable de la plate-forme.

1.1 Outils utilisés

La reprise du projet s'est fait avec le gestionnaire de version (Redmine) de polytech, dans la suite du projet le même outil sera utilisé pour gérer l'évolution du projet, côté développeur TortoiseSVN et l'outil utiliser pour communiquer avec le gestionnaire de version.

Afin de faciliter le suivi des tâches, Trello qui est un outil de gestion de projet en ligne, où l'on peut définir des tâches et les regrouper en fonction de leur nature et de leur avancement, cet outil a aussi permis avec l'encadrant des échanges d'information, comme par exemple, des documents rapports, liens sur des documents scientifiques, Thèses.

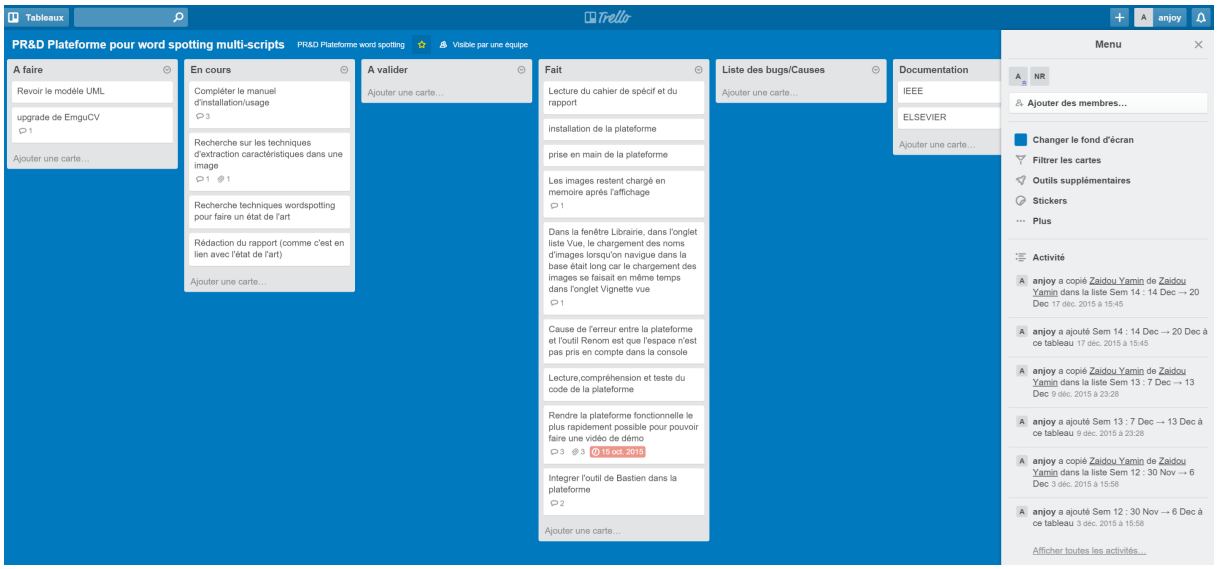


Figure 1 – *Capture d'écran de Trello*

2 Gestion des réunion et rapport hebdomadaire

Avec l'outil Trello, pour faire suivre mon travail à Monsieur N. Ragot, j'ai mis en place des rapports hebdomadaires de type de l'image suivante :

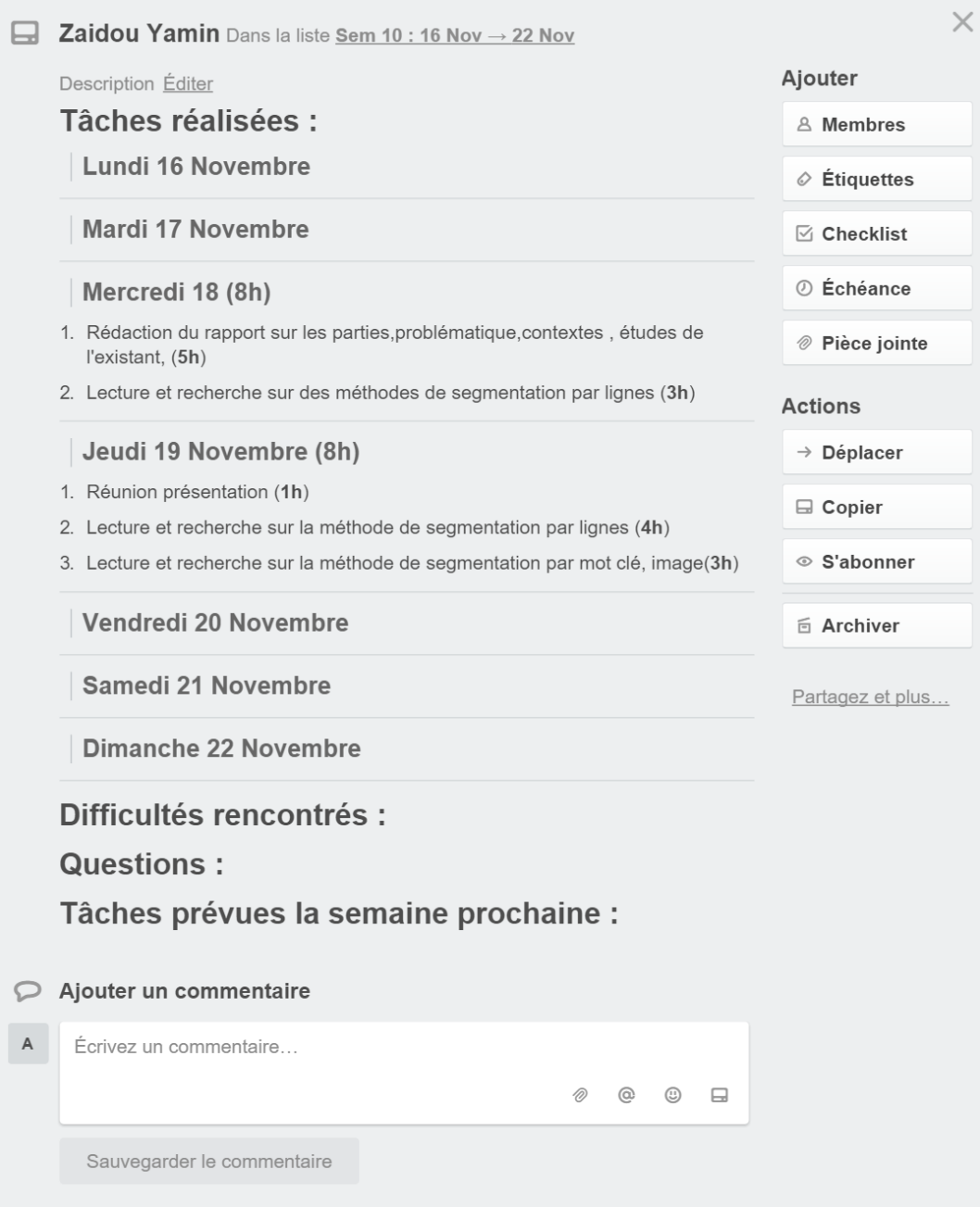


Figure 2 – *Capture d'écran de Trello*

Dans ce rapport hebdomadaire est présenté le travail effectué dans la semaine celui prévu dans celle d'après, les questions et problèmes rencontrés.

Pour cette première partie des réunions de présentation ont été effectuées à chaque fin de réalisation de tâches. Aussi des petites réunions pendant les pauses...

9 Bilan de la partie développement

La deuxième partie de ce projet a été plus qu'enrichissante, elle m'a permis de découvrir beaucoup de nouvelles techniques de développement, d'approfondir mes connaissances en c# et dans la gestion des projets. J'ai eu grâce à ce projet, une manière différente de développer en pensant toujours à avoir une meilleure qualité de code, respect des conventions de codage, documentation, tests unitaires fonctionnels, utilisation des outils de projets comme Jenkins, sonar, resharper.

J'ai eu la chance de gérer dans ce projet plusieurs petits projets très intéressants les unes des autres, la possibilité de réfléchir et les concevoir, de les associer en respectant la chaîne d'exécution pour avoir un résultat.

L'outil de segmentation était comme un défi et ayant proposé la méthode de segmentation. Voir des résultats très positifs était très satisfaisant.

En général même s'il reste beaucoup d'amélioration pour les outils développer ainsi qu'à la plate-forme, les tâches demandées ont été accomplis et il s'agit pour ma part du projet le plus intéressant de toute ma formation.

Comptes rendus hebdomadaires

Compte rendu n°1 du 14/09/2015

Mercredi 16 Septembre (7H)
1. Réunion et présentation de la salle RFAI (0.5H)
2. Lecture du cahier de spécif (Loreen) (1.5H)
3. Réunion avec l'encadrant (1 H)
4. Lecture de tuto SVN (1 H)
5. Installation de machines virtuelle (0.5)
6. Lecture du rapport de Loreen (2 H)
7. lecture des digrammes UML (0.5)

Jeudi 17 Septembre (8H)
1. Importation du projet avec les DLL (0.5H)
2. Lecture bref du code et petite prise en main de visual (1H)
3. Comparaison des interfaces avec le rapport (0.5H)
4. Comparaison du code avec le modèle UML (2H)
5. Importation des images et utilisation de la plateforme (0.5H)
6. Test de la plateforme et détection des bugs (2H)
7. Lecture et compréhension du code (1.5H)

Compte rendu n°2 du 21/09/2015

Mercredi 25 Septembre (8H)
1. Lecture du code et test de la plateforme (2H)
2. Lister des bugs et trouver les causes (2 h)
3. Lecture du code et test de l'outil renom (2H)
4. Debug et correction de certains bugs (2H)

Jeudi 26 Septembre (5H)
1. Correction de Bugs dans l'interface (1H)
2. Lecture, compréhension de l'architecture et le code pour intégrer l'outil de Bastien (4h)

Compte rendu n°3 du 28/09/2015

Mardi 29 Septembre (1.5H)
1. Lecture, compréhension de l'architecture et le code pour intégrer l'outil de Bastien (1.5h)

Mercredi 30 Septembre (8H)
1. Lecture du code et compréhension des fonctions (partie utilisation des outils (2H)
2. Débogage de l'application pour comprendre et recréer l'architecture de l'outil de Bastien (4h)
3. Correction des bugs de l'application (2H)

Jeudi 01 Octobre (8H)
1. Lecture du code et compréhension des fonctions (partie utilisation des outils (2H)
2. Débogage de l'application pour comprendre et recréer l'architecture de l'outil de Bastien (3h)
3. Création du cahier de spécification (plan de développement) découpage du projet en tache et planification (3h)

Compte rendu n°4 du 05/10/2015

Mardi 06 Octobre (2h)
1. Teste et création la base pour utiliser l'outil de Bastien. (2h)

Mercredi 07 Octobre (8h)
1. Lecture du rapport de Bastien pour pouvoir créer les fichiers (Cdp) (1.5h)
2. Teste et création la base pour utiliser l'outil de Bastien (6h)
3. Installation de logiciel et test pour faire une vidéo sous Windows (0.5h)

Jeudi 08 Octobre (8h)
1. Teste et création la base pour utiliser l'outil de Bastien (4h)
2. Faire la vidéo de démonstration de l'application (2h)

Compte rendu n°5 du 12/10/2015

Mercredi 14 Octobre (8h)
1. Traduire l'application en anglais (1h)
2. Refaire la vidéo avec les corrections demandées (2h)
3. Compléter le cahier de spécification sur le plan de développement (3h)
2. Compléter le manuel d'installation et d'utilisation (2h)

Jeudi 15 Octobre (8h)
1. Refaire la vidéo avec les corrections demandées (0.5h)
2. Lecture du rapport sur le projet science de la décision (0.5h)
3. Lire et comprendre le code du projet science de la décision (1h)
4. Chercher et comprendre les méthodes d'extraction de caractéristiques (pour l'état de l'art du projet) (6h)

Compte rendu n°6 du 19/10/2015

Mercredi 21 Octobre (8h)
1. Lecture du code et du rapport (Science de la décision) (2h)
2. Recherche et lecture des documents, rapports, techniques de word-spotting pour la partie état de l'art (6h)

Jeudi 22 Octobre (7h)
1. Recherche des documents,rapports, techniques de word spotting pour la partie état de l'art (2h)
2. Pareil mais sur les techniques d'extraction de caractéristiques sur une voient les avantages et

inconvenants de chacun (5h)

Compte rendu n°7 du 26/10/2015

Mercredi 27 Octobre (2h)
1. Recherche et lecture de document pour l'état de l'art (2h)

Vendredi 30 Octobre (2h)
1. Recherche et lecture de document pour l'état de l'art (2h)

Dimanche 1 Novembre (2h)
1. Lecture de document pour l'état de l'art (2h)

Compte rendu n°8 du 02/11/2015

Mercredi 4 Novembre (8h)
1. Recherche et lecture de document pour l'état de l'art (3h)
2. Rédaction du rapport (5h)

Compte rendu n°9 du 09/11/2015

Mercredi 11 (4h)
1. Lecture des documents fournis (2h)
2. Rédaction du rapport (2h)

Jeudi 12 Novembre (6h)
1. Rédaction du rapport (3h)
2. Recherche et lecture des documents sur les techniques de Words Spotting (3h)

Compte rendu n°10 du 16/11/2015

Mercredi 18 (8h)
1. Rédaction du rapport sur les parties,problématique,contextes , études de l'existant, (5h)
2. Lecture et recherche sur des méthodes de segmentation par lignes (3h)

Jeudi 19 Novembre (8h)
1. Réunion présentation (1h)
2. Lecture et recherche sur la méthode de segmentation par lignes (4h)
3. Lecture et recherche sur la méthode de segmentation par mot-clé, image(3h)

Compte rendu n°11 du 23/11/2015

Mercredi 25 (8h)
1. Lecture et recherche sur des méthodes de segmentation par lignes (6h)
- Projection–based methods (plus précisément sur cette méthode)
- Smearing methods
- Grouping methods
2. Lecture et Recherche sur des méthodes de segmentation par mot-clé (2h)

Jeudi 26 Novembre (2h)
1. Recherche sur des méthodes de segmentation par mot clé (2h)

Compte rendu n°12 du 30/11/2015

Mercredi 2 Décembre (8h)
1. Lecture et recherche sur des méthodes de segmentation par mot ligne (6h)
- Hough transformation
- Projection–based methods
- Smearing methods
- Grouping methods
2. Lecture et Recherche sur l'extraction des caractéristiques (2h)

Jeudi 3 Décembre (8h)
1. Réunion présentation (1h)
2. Synthèse des documents et information recherchée (4 h)
3. Recherche approfondie sur les 3 algo principaux (3h)
- Hough transformation
- Projection–based methods
- Smearing methods

Compte rendu n°13 du 07/12/2015

Mercredi 9 Décembre (8h)
1. Lecture et Recherche sur l'extraction des caractéristiques (4h)
- Method ColOmn base
- HOG
2. Rédaction du rapport (4h)

Jeudi 10 Décembre (8h)
1. Recherche des programmes sur les méthodes de segmentation pour des tests et comparaison (6h)
2. Rédaction du rapport (2h)

Compte rendu n°14 du 14/12/2015

Mercredi 16 Décembre (8h)
1. Rédaction du rapport (6h)
2. Recherche des programmes sur les méthodes de segmentation pour des tests et comparaison (2h)

Jeudi 17 Décembre (7h)
1. Rédaction du rapport (7h)

Compte rendu n°15 du 21/12/2015

mardi 22 Décembre (3h)
1. Rédaction du rapport (3h)
mercredi 23 Décembre (2h)
1. Rédaction du rapport (2h)

Compte rendu n°16 du 28/12/2015

mardi 29 Décembre (3h)
1. Rédaction du rapport (3h)
mercredi 30 Décembre (1h)
1. Rédaction du rapport (1h)
vendredi 01 Janvier (1h)
1. Rédaction du rapport (1h)
samedi 02 Janvier (2h)
1. Rédaction du rapport (2h)

dimanche 03 Janvier (5h)
1. Rédaction du rapport (5h)

Webographie

[WWW1] *Document Image Binarization and binarization Algorithm*. URL : https://github.com/zp-j/binarizewolfjolion#fca-resources (visité le 20/03/2016).

[WWW2] *Features for Word Spotting in Historical Manuscripts*. URL : http://ciir.cs.umass.edu/pubfiles/mm-39.pdf (visité le 22/03/2016).

[WWW3] *Segmentation-based Historical Handwritten Word Spotting using Document-Specific Local Features*. URL : https://lib-repos.fun.ac.jp/dspace/bitstream/10445/3001/4/kterasaw_2009_01_icdar116.pdf (visité le 02/01/2016).

[WWW4] *Text Line Segmentation of Historical Documents : a Survey*. URL : http://arxiv.org/ftp/arxiv/papers/0704/0704.1267.pdf (visité le 03/12/2015).

Bibliographie

[1] Loreen LAMBIN. *Cahier de spécification système*. Rapp. tech. Plate-forme de Word-Spotting. France : École Polytechnique de l'Université de Tours, jan. 2015.

[2] Loreen LAMBIN. « Rapport final de projet de fin d'études Plate-forme de Word-Spotting ». Mém.de mast. France : UNIVERSITÉ FRANÇOIS - RABELAIS DE TOURS, mai 2015.

[3] Tanmoy MONDAL. « From Time Series Signal Matching to Word Spotting in Multilingual Historical Document Images ». Mém.de mast. France : UNIVERSITÉ FRANÇOIS - RABELAIS DE TOURS, déc. 2014.

[4] Nazih Ouwayed. « Segmentation en lignes de documents anciens :application aux documents arabes ». Mém.de mast. France : Universite Nancy II, juin 2010.

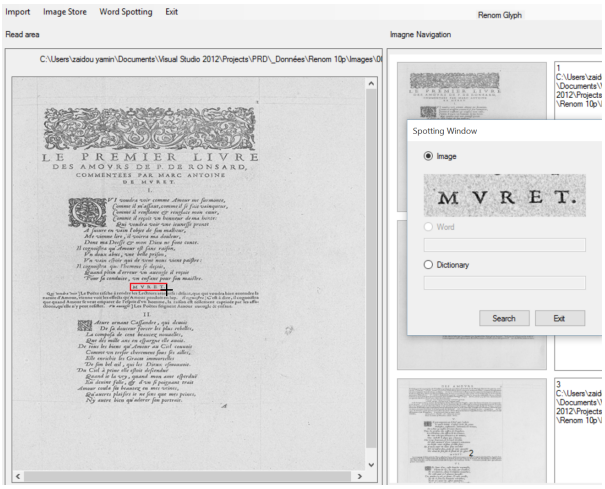
Plateforme pour word spotting multi-scripts : Plateforme pour word spotting multi-scripts

Yamin Zaidou
Encadrement : Nicolas Ragot

En collaboration avec
Laboratoire Informatique

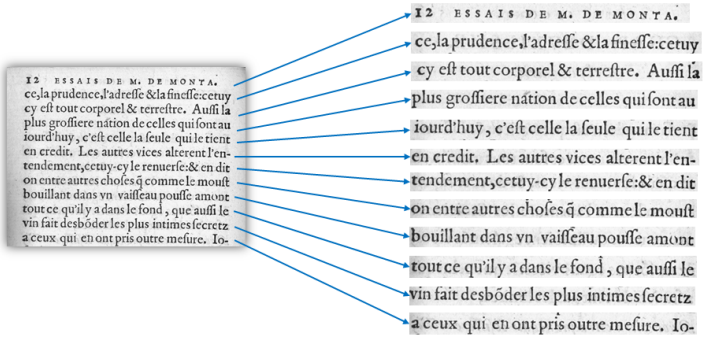
Plateforme Word soppting

Une plateforme de word spotting permet à des utilisateurs de pouvoir effectuer une recherche dans des documents anciens et manuscrits. La plateforme du laboratoire Informatique de Polytech Tours effectue une recherche en comparant des images décrit par des caractéristiques et non pas par le contenue textuel comme les applications OCR.



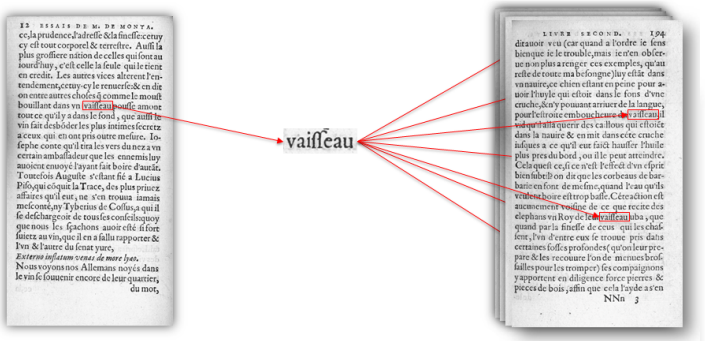
Objectifs

Faire évoluer la plateforme de Word Spotting : pour prendre en compte la segmentation par mot/ligne, pour être capable d'extraire des caractéristiques sur une image et Améliorer le code pour faciliter sa maintenabilité et l'interface. Tout en gardant une version stable de la plateforme. Ces objectifs permettront à l'application d'être autonome et pouvoir utiliser les outils de word spotting.



Résultats attendus

Une plateforme capable de : Segmenter en ligne/mot une base d'image, extraire les caractéristiques d'une image et avoir une interface intuitive et facile d'utilisation.



Ecole Polytechnique de l'Université François Rabelais de Tours
Département Informatique
64 avenue Jean Portalis, 37200 Tours, France
www.polytech.univ-tours.fr

POLYTECH TOURS
Informatique

Plateforme pour word spotting multi-scripts

Plateforme pour word spotting multi-scripts

Résumé

Ce document a pour but de décrire en deux parties le projet de recherche et de développement, encadré par M. ragot. Une première partie consacrée à la recherche sur la segmentation des documents anciens, l'étude de l'existant et une analyse/conception pour corriger, améliorer l'application word-spotting.

La deuxième partie concerne la description de tous les développements effectuer. Une présentation des méthodes utilisées et leurs résultats. Les améliorations effectuées par rapport au projet précèdent

Mots-clés

word spotting, Matching, plate-forme, User Control, HOG

Abstract

This document aims to describe in two parts my research developpement project under M. Ragot. In this document, we will talk about the studies on the segmentation methods of olds document. Then we will present the previous projects on this subject. At thé end, we will show all the analyses/conceptions to correct and upgrade the word-spotting application.

The second part concerns the description of all the developments done. A presentation of the methods used and their results. The improvements made compared to the previous project

Keywords

word spotting, Matching, plateform, User Control, HOG

Entreprise
Laboratoire Informatique

Tuteurs entreprise

Étudiants
Yamin Zaidou (DI5)

Tuteurs académiques
Nicolas Ragot