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
polytech.univ-tours.fr

Projet Recherche & Développement

2016-2017

Reconnaissance d'éléments textuels ou graphiques dans les documents juridiques numérisés

Entreprise
FoxNot

Tuteurs entreprise
Thierry Arnaly
Dominique Bazin

Étudiant
Horacio Cachinho (DI5)

Tuteurs académiques
Jean-Yves Ramel
Mohamed Slimane

2 avril 2017

Liste des intervenants

Entreprise

FoxNot
45 Rue de Buffon
37000 TOURS
www.foxnot.com

Nom	Email	Qualité
Horacio Cachinho	horacio.cachinho@etu.univ-tours.fr	Étudiant DI5
Jean-Yves Ramel	jean-yves.ramel@univ-tours.fr	Tuteur académique, Département Informatique
Mohamed Slimane	mohamed.slimane@univ-tours.fr	Tuteur académique, Département Informatique
Thierry Arnaly	thierry.arnaly@foxnot.com	Tuteur entreprise
Dominique Bazin	dominique.bazin@foxnot.com	Tuteur entreprise

Avertissement

Ce document a été rédigé par Horacio Cachinho susnommé l'auteur.

L'entreprise FoxNot est représentée par Thierry Arnaly et Dominique Bazin susnommés les tuteurs entreprise.

L'Ecole Polytechnique de l'Université François Rabelais de Tours est représentée par Jean-Yves Ramel et Mohamed Slimane susnommés les tuteurs académiques.

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 des tuteurs académiques 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

Horacio Cachinho, *Reconnaissance d'éléments textuels ou graphiques dans les documents juridiques numérisés*, Projet Recherche & Développement, Ecole Polytechnique de l'Université François Rabelais de Tours, Tours, France, 2016-2017.

```
@mastersthesis{
    author={Cachinho, Horacio},
    title={Reconnaissance d'éléments textuels ou graphiques dans les documents juridiques
      numérisés},
    type={Projet Recherche \& Développement},
    school={Ecole Polytechnique de l'Université François Rabelais de Tours},
    address={Tours, France},
    year={2016-2017}
}
```

Document mis en forme sous LaTeX
grâce à la classe de document `polytech.cls` V2.2.4 (2017/03/28)

Table des matières

Table des figures

Liste des Algorithmes

1 Introduction

La rédaction des actes notariés est une tâche qui peut s'avérer longue et difficile. Une grande quantité d'informations entre en jeu lors de la mise en place de ces actes, et l'échange et traitement de ces données peut très vite s'avérer délicate. Aujourd'hui, tous les documents nécessaires à l'établissement des actes notariés sont encore traités manuellement ce qui ralenti le processus et allonge les délais de création des dossiers.

FoxNot propose désormais un service qui permet de dématérialiser ces échanges et faciliter la mise en place des actes notariés. Grâce à ce service, un particulier peut trouver et contacter un notaire rapidement, échanger des documents facilement et suivre l'état de son dossier en temps réel.

Cependant, même si une avancée à été effectuée au niveau de la création du dossier et des échanges de documents, ces derniers sont encore traités manuellement, tâche qui, comme mentionné précédemment, peut s'avérer longue et fastidieuse, autant pour le notaire que pour le particulier concerné. De plus, un grand nombre de données sont redondantes sur la plus part des documents (identification, adresse, numéros de téléphone, etc) ce qui rend le traitement de ces documents parfois inefficace, sans oublier les possibles fautes de saisie, qui résulteront dans un délai d'aboutissement du dossier encore plus important.

Ce projet de recherche et développement, qui se déroulera en deux étapes, aura pour but de trouver une solution efficace permettant la reconnaissance et l'extraction de données d'intérêt (sous forme textuelle ou graphique) sur des documents juridiques préalablement dématérialisés (scan ou photo). Lors de la première étape, celle de recherche, on étudiera les technologies existantes sur la détection et lecture de documents dématérialisés afin de définir les méthodes les plus adaptées et efficaces à la résolution de notre problématique, cela au travers de l'étude du contexte et de la réalisation d'un état de l'art. La deuxième étape, celle du développement, permettra de mettre en place la technologie définie lors de l'étape de recherche, au moyen d'un *framework* exploitable par l'entreprise.

Première partie

Recherche

2 Description générale

1 Contexte

Lors de la rédaction d'actes notariés, un notaire a besoin d'un grand nombre d'informations et de documents relatifs aux personnes et/ou biens concernés. Ainsi, de part leur quantité et complexité, leur traitement peut s'avérer rapidement long et peut être sujet à des erreurs de saisie et/ou compréhension.

C'est dans ce contexte que FoxNot cherche à implémenter une solution permettant le traitement des documents de manière automatisée, afin de faciliter la récupération des données d'intérêt.

2 Description du problème

Lorsqu'un particulier lance la création d'un dossier auprès d'un notaire, ce dernier nécessite un certain nombre de documents juridiques (carte d'identité, passeport, titre de propriété, etc) car ces documents possèdent des informations qui sont primordiales pour la mise en place du dossier.

Actuellement, certaines de ces informations sont saisies par le particulier via les formulaires proposés par le service de FoxNot, afin de faciliter et accélérer le traitement de la demande. Cependant, la saisie de nombreuses informations peut s'avérer longue et devient vite sujette à des erreurs de saisie, qui impactent grandement la durée de mise en place du dossier, et auront ainsi un effet inverse sur l'objectif principal du système.

3 Environnement du projet

Ce projet vise à créer un module de reconnaissance de documents qui sera intégré au système actuel de FoxNot, et permettra d'apporter une solution au problème étudié.

Le système dans lequel le module de reconnaissance de documents devra s'intégrer fonctionne de la manière suivante :

- Tout d'abord le client se connecte au site web de FoxNot. De part ses manipulations sur le site, une requête est envoyée au serveur *RabbitMQ* de FoxNot.
- Ce serveur permet la gestion de files d'attente contenant des *messages* destinées aux applications existantes dans le système. Ces *messages* peuvent contenir des informations relatives à un processus à exécuter, ou des données à échanger entre deux applications. Une fois cette requête reçue par le serveur *RabbitMQ*, le *message* associé va être dirigé par l'échangeur vers la file d'attente correspondante.
- Chaque file d'attente a sa finalité. Une application peut *souscrire* à une ou plusieurs files et récupérer les messages qui y sont stockés. Il y aura donc une file destinée aux messages de traitement des documents, à laquelle le module de reconnaissance *souscrira*. Les messages destinés à ce module y seront stockées et traités de manière FIFO.
- Dés lors qu'un *message* est stocké dans le file, le module le récupère et effectue le traitement associé.
- Les messages étant génériques, *RabbitMQ* permet la communication entre modules développés dans des langages différents. Malgré que le système actuel soit développé en PHP, le module de reconnaissance des documents peut être développé en C# et profiter des nombreuses librairies de traitement existantes pour ce langage.

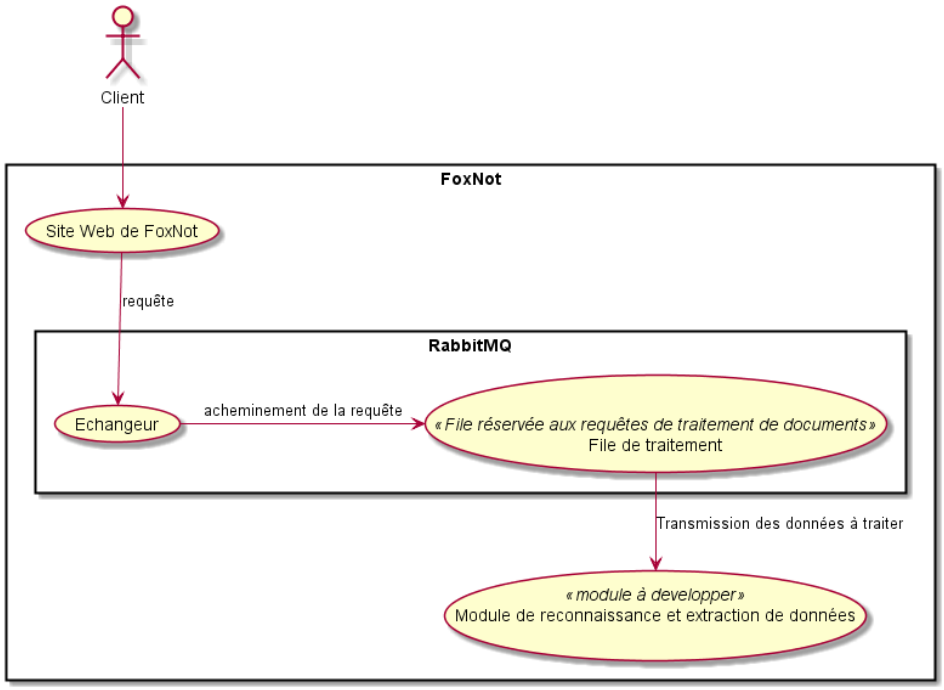


Figure 1 – *Environnement du projet*

Au départ, le module de reconnaissance de documents devra fonctionner de manière autonome. L'intégration au système existant de FoxNot ne s'effectuera que plus tard dans le projet, si le temps le permet.

4 Caractéristiques des utilisateurs

Les *workers* développés seront manipulés indirectement par les utilisateurs du service proposé par FoxNot. Ces utilisateurs n'auront besoin d'aucune compétence informatique spécifique puisque le système sera complètement autonome et transparent pour eux.

5 Architecture générale du système

Le module de reconnaissance de documents à développer pendant ce projet sera constitué de *workers*. Ces *workers* permettront de séparer les tâches au sein du module.

D'après les objectifs définis plus tôt, il y aura 6 *workers* au sein du module à développer :

- *Worker* de chargement et pré-traitement des images
  - Ce *worker* devra être capable de charger une image (scan ou photo) de manière à ce que les *workers* d'extraction des caractéristiques et de lecture par OCR soient capables de la traiter. C'est donc ce *worker* qui s'occupera de tout pré-traitement de l'image (binarisation, recadrage, redressage, netteté) si nécessaire.
- *Worker* chargé de l'extraction des caractéristiques
  - Ce *worker* comportera l'implémentation des algorithmes d'extraction des caractéristiques et les liaisons aux librairies nécessaires. L'objectif est que ce *worker* soit capable de prendre une image quelconque contenant un document, et d'extraire ses caractéristiques (sous forme d'un vecteur).
- *Worker* chargé de la classification du document
  - Ce *worker* devra être capable de comprendre le vecteur de caractéristiques généré par le *worker* d'extraction et de pouvoir définir la classe du document associé grâce à une base d'apprentissage (base d'images utilisée par les algorithmes de classification). Pour cela, ce *worker* comportera les liaisons aux librairies d'algorithmes de classification (OC-KNN [1], K-NN [2], SVM [3]).
- *Worker* chargé de la lecture par OCR
  - Ce *worker* sera en charge de lancer une lecture OCR sur le document. Pour cela il sera lié à un système d'OCR (tesseract) et devra être capable d'interpréter la sortie de ce système. Cette sortie prendra la forme d'un fichier de texte brut contenant toutes les données lues par le système d'OCR sur le document.
- *Worker* chargé de l'analyse et extraction sémantique
  - Ce *worker* sera en charge d'analyser sémantiquement le fichier de données généré par le *worker* de lecture par OCR. Pour cela il va prendre en considération la classe du document traité (fournie par le *worker* de classification) et une liste des champs à détecter (nom, prénom, adresses entre autres). Par la suite, une analyse des données brutes sera effectuée et si un champs est détecté, la donnée correspondante est extraite dans un fichier *XML*. Suite à l'analyse, le fichier *XML* contenant l'ensemble des données d'intérêt extraites est sauvegardé afin d'être exploité par la suite par une autre application.
- *Worker* chargé de la l'apprentissage de nouveaux cas d'utilisation
  - Ce *worker* permettra d'enseigner le système à détecter de nouveaux documents. Il permettra ainsi la création de nouvelles classes et d'ajouter des images à la base d'apprentissage.

Le diagramme suivant démontre les interactions possibles entre les acteurs et le module à développer :

Le diagramme en annexe démontre l'architecture générale du système. Chaque *worker* sera détaillé par la suite.

1. One-class K-NN
2. K-Nearest Neighbors ou K Plus proches voisins
3. Support Vector Machine ou Machine à vecteurs de support

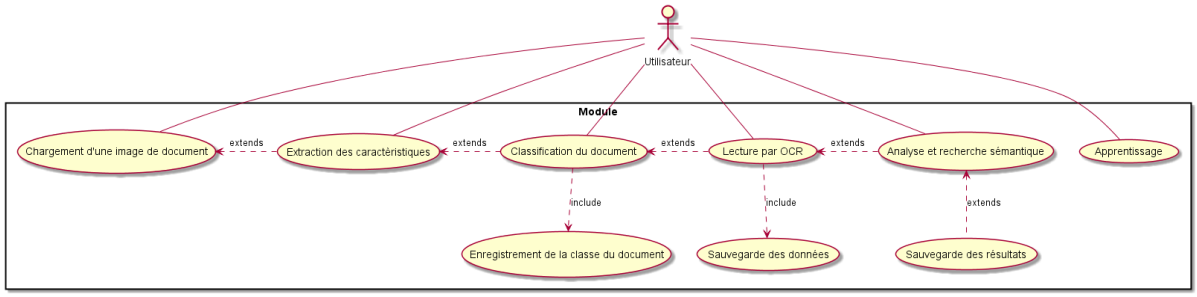


Figure 2 – *Interactions fonctionnelles entre les acteurs et le système*

6 Objectifs

L'objectif de ce projet est de créer un module composé d'un ensemble de *workers* capables de répondre aux problématiques suivantes :

1. Prise en compte d'une image (numérisée ou photo) contenant un document juridique et capacité à classifier ce document (reconnaître son type), malgré les éventuelles distorsions qu'une numérisation peut engendrer (rotation, échelle, résolution, netteté). Les documents à prendre en compte sont :
   - Cartes d'identité françaises
   - Passeport de type français
   - Diagnostic de performance énergétique (DPE), utilisé pour la validation du POC
   - — Les documents mentionnés ci-dessus seront la base de tests lors des développements. L'objectif sera que le système soit capable d'évoluer afin de permettre de détecter plus de documents dans le long terme.
2. Lecture automatique d'un document par OCR et extraction des données contenues dans ce même document sous forme de fichier texte.
3. Analyse et recherche sémantique sur les données extraites par le module d'OCR. Cette analyse sémantique permettra la détection des données d'intérêt à extraire (nom, prénom, adresse, numéro de téléphone, *entre autres*).
4. Module d'apprentissage générique, qui permettra l'évolution du système dans le long terme. Ce module permettra l'apprentissage de nouvelles classes de documents basés sur un ensemble de documents de même type (base d'apprentissage).

Les problématiques [2] et [3] mentionnés ci-dessus devront être résolues de manière à pouvoir s'adapter au document pris en charge par le système. Ainsi, les traitements à effectuer et les données à extraire ne sont pas les mêmes si le document s'agit d'une carte d'identité ou bien d'un passeport.

7 Contraintes

7.1 Contraintes de développement et conception

7.1.1 Langages de programmation

Le système sera à développer avec le langage C# et le *framework* .NET, dans ses versions 6 et 4.6+ respectivement.

7.1.2 Environnement et Logiciels

Le système à développer repose sur les outils suivants :

- Distribution Linux Debian 8 (jessie) 64 bits
- Serveur *RabbitMQ* (3.6.5)
- *Docker* (version 1.11.1, build 5604cbe)
- Système de *versionning Git*
- *Mono* et *MonoDevelop*, qui seront requis afin de permettre la bonne exécution du *framework* .NET sous Linux.

7.1.3 Contraintes de développement

Le développement du système devra se dérouler en 3 étapes :

1. : Reconnaissance d'un document à partir d'une image
2. : Lecture et analyse d'un document par OCR
3. : Analyse et recherche sémantique des données extraites par l'outil d'OCR

7.2 Contraintes de fonctionnement et d'exploitation

7.2.1 Performances

D'un point de vue utilisateur, l'application devra être la plus réactive possible. Dans la mesure où le traitement d'images peut devenir coûteux en temps (les images scannées sont généralement de très grande taille), un temps de latence est toujours à prévoir. Le but sera donc de diminuer ce temps de traitement au maximum.

3 État de l'art

Ici seront présentées les technologies et méthodologies existantes sur la reconnaissance et la lecture de documents dématérialisés.

1 Algorithmes de pré-traitement

1.1 Filtre médian

Le filtre médian [9] est, comme son nom indique, un filtre numérique permettant le lissage et la réduction du bruit sur une image. Pour chaque pixel d'une image, le filtre va ainsi calculer la valeur médiane de son voisinage et remplacer la valeur du pixel par cette valeur. L'illustration suivante tirée de [WWW34] détaille le fonctionnement du filtre :

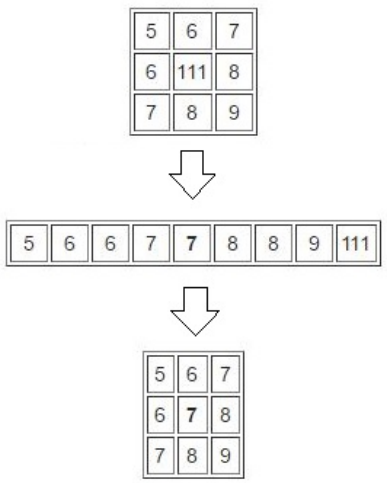


Figure 1 – *Fonctionnement du filtre médian*

Voici un exemple d'image bruitée à laquelle un filtrage médian 3x3 à été appliqué :

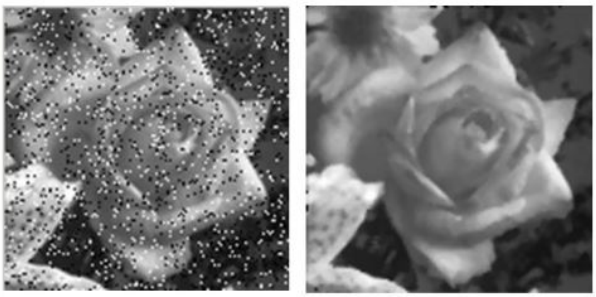


Figure 2 – *Application d'un filtrage médian 3x3 sur une image bruitée*

1.2 Binarisation

La binarisation consiste à convertir les pixels d'une image en pixels binaires (soit noirs, soit blancs). Dans [11] les auteurs présentent une méthode de binarisation adaptative. Cette méthode prends en compte la texture de l'image à binariser pour décider du meilleur algorithme à utiliser. Le but est ainsi de calculer le meilleur seuillage pour chaque pixel permettant d'obtenir une image binarisée de la meilleur qualité possible.

Figure 3 – *Exemple de binarisation*

La binarisation est particulièrement importante dans le domaine de l'extraction de données par OCR car cela facilite la lecture du moteur et améliore par la même occasion les taux de reconnaissance des éléments.

1.3 Redressement des images

Une des limites des moteurs d'OCR est l'angle de lecture du texte à analyser. En effet, plus le texte est incliné et moins efficace sera sa reconnaissance. Les documents à traiter venant de scanners ou d'appareils photo, l'angle de prise est une des problématiques les plus importantes à traiter. Afin d'obtenir les meilleurs résultats, l'image devra ainsi être redressée (si besoin) avant d'être analysée par le moteur d'OCR.

Dans [10] les auteurs listent les méthodes existantes pour la détection d'inclinaison d'une image ainsi que les avantages et inconvénients de chacune. Parmi les méthodes listées on note :

— **Transformée de Hough :** se base sur la détection de lignes (alignement des pixels). Cette technique est néanmoins très complexe et sans optimisation, la méthode est inutilisable pour des images de grande taille.
— **La projection des profils :** la méthode se base dans la projection horizontale et verticale des pixels noirs et crée des histogrammes de projection. Des rotations sont ensuite effectuées jusqu'au moment ou les projections horizontales sont les plus élevées. La méthode consomme néanmoins beaucoup de ressources CPU du au calcul intensif des histogrammes.

— **Nearest-Neighbor Clustering :** cette méthode se base sur la cohérence des alignements des blocks de caractères. Les caractères ascendants et descendants (q, p, T, l...) créent néanmoins des blocs qui ne sont pas alignés ce qui fausse la détection, même sur une image droite. La méthode est également très sensible au bruit.

Les moteurs d'OCR comme tesseract ou FineReader proposent néanmoins des méthodes optimisées de redressement.

Figure 4 – *Exemple d'image scannée avec angle d'inclinaison*

2 Algorithmes d'analyse structurelle

2.1 RLSA (Run Length Smoothing Algorithm)

Le RLSA est un algorithme de segmentation qui à pour but de créer des zones permettant d'identifier les différentes composantes d'un document. Basé sur l'explication de Olivier Augereau dans [WWW3], le RLSA crée ces zones en reliant les pixels noirs (*foreground*) qui sont séparés par des pixels blancs (*background*). La quantité de pixels blancs minimale (n) pour créer la séparation est définie par rapport au détail de segmentation : plus n est petit, plus il y aura de zones segmentées (séparation lettre à lettre) et plus n est grand moins il y aura de zones segmentées (séparation paragraphe à paragraphe).

L'algorithme s'applique de manière horizontale puis verticale (les espacements horizontaux et verticaux n'ayant, d'habitude, pas la même taille) puis les résultats sont combinés par ET logique [WWW18]. Cependant, cet algorithme n'est applicable que sur des documents ayant une structure de type *Manhattan*.

2.2 RLSO (Run Length Smoothing with OR)

L'algorithme RLSO est un algorithme basé sur RLSA mais qui est destiné aux documents ayant une structure de type *Non-Manhattan*. Dans [4] les auteurs expliquent le fonctionnement de l'algorithme. RLSO reste assez similaire à RLSA mais utilise le OU logique (plutôt que le ET logique) ce qui permet de préserver les pixels noirs de l'image originelle et ne requiert donc pas une deuxième passe horizontale pour rétablir la structure d'origine. Une autre amélioration apportée sur cet algorithme est la définition automatique du seuillage, basée sur la distribution des espacements dans le document.

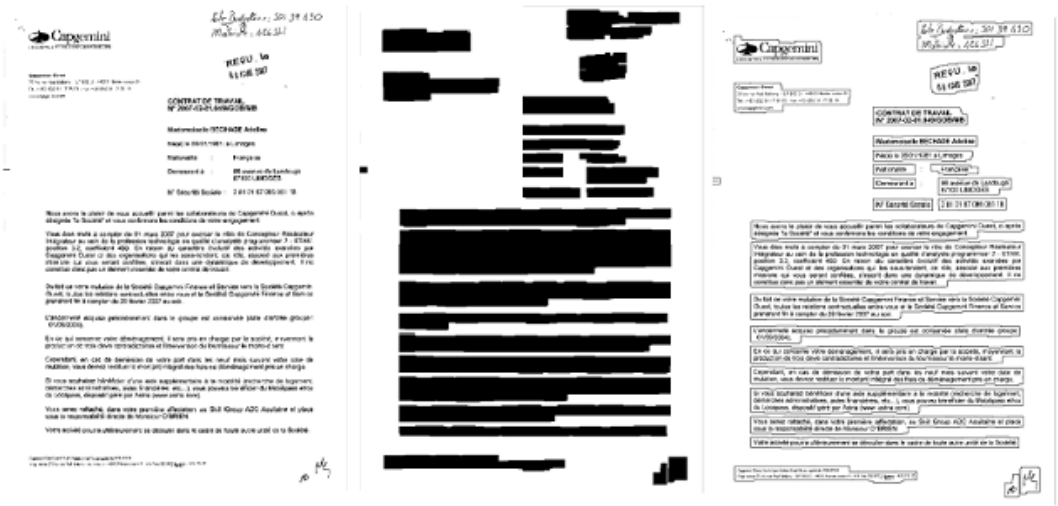


Figure 5 – *Exemple d'application de RLSA sur un document*

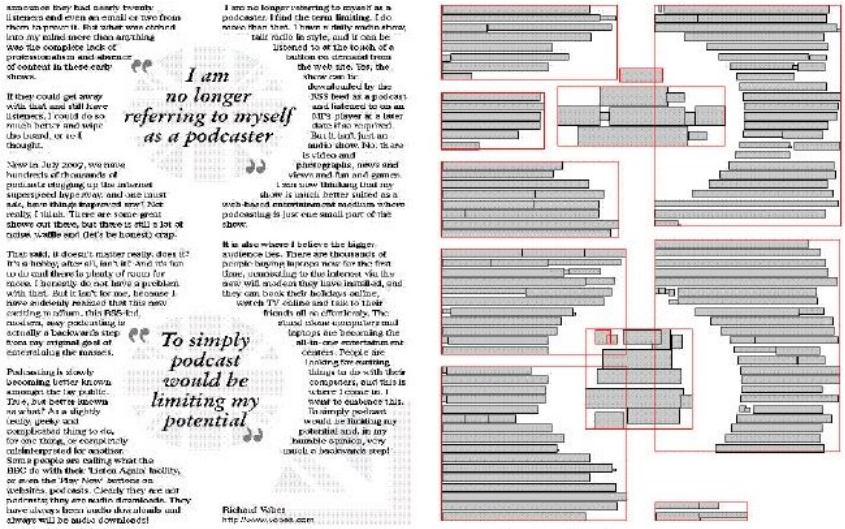


Figure 6 – *Exemple d'application de RLSO sur un document*

3 Algorithmes d'extraction de caractéristiques et descripteurs

3.1 Algorithmes d'extraction de caractéristiques

3.1.1 SIFT

SIFT ou *Scale Invariant Feature Transform* est un algorithme utilisé pour détecter et identifier des fragments identiques entre deux images. Comme expliqué dans [6], l'algorithme est divisé en deux parties : détection et calcul des descripteurs puis mise en correspondance. SIFT commence ainsi par calculer des descripteurs (les descripteurs SIFT) qui vont permettre de caractériser le contenu de l'image de manière indépendante de l'échelle, de la résolution, de la rotation ainsi que de la netteté. La détection s'effectue sur des zones circulaires de rayon *"facteur d'échelle"* (x, y, σ), centrées au tour d'un point d'intérêt de l'image. Cela mène à une construction d'un histogramme des orientations locales des contours puis l'histogramme est mis sous forme d'un vecteur à 128 dimensions. Ce descripteur est connu pour être robuste et est utilisé dans plusieurs

domaines comme la détection de points d'intérêt, la mise en correspondance d'images, le suivi d'objets ou l'assemblage d'images.

Figure 7 – *Points d'intérêt détectés par SIFT*

Suite au calcul des descripteurs, SIFT effectue une mise en correspondance avec une base d'images de référence utilisant la distance euclidienne. Parmi toutes les correspondances effectuées par l'algorithme, des sous-ensembles sont définis basés sur la qualité de la correspondance puis les meilleurs sont conservés. Un modèle probabiliste est enfin appliqué afin de confirmer cette correspondance.

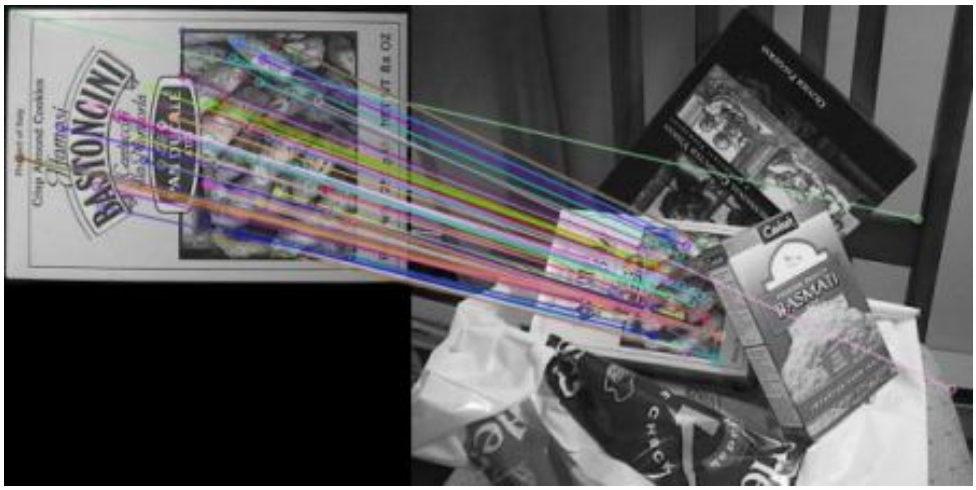


Figure 8 – *Mise en correspondance de deux images par SIFT*

3.1.2 SURF

SURF ou *Speeded Up Robust Features* est un algorithme, en partie basé sur SIFT, mais plus rapide et robuste que ce dernier. Sa rapidité est due à l'utilisation d'images intégrales et le calcul des points d'intérêt effectué par approximation de matrice Hessienne. L'algorithme est expliqué en

détail dans [3].

D'après le site officiel [WWW28] l'algorithme n'est pas utilisable pour des fins commerciales.

Dans [WWW4], Olivier Augereau fait une comparaison entre les deux algorithmes SIFT et SURF :

MÉTHODE	TEMPS	ÉCHELLE	ROTATION	FLOU	ILLUMINATION	AFFINE
SIFT	normal	meilleur	meilleur	meilleur	normal	bon
PCA-SIFT	bon	normal	bon	normal	bon	bon
SURF	meilleur	bon	normal	bon	meilleur	bon

Figure 9 – *Tableau comparatif SIFT / SURF*

Cependant, d'après Olivier Augereau dans [2], SIFT ou SURF ne sont pas tout à fait adaptées aux images de documents du fait de leur mauvaise détection de patterns répétitifs ou d'images faiblement texturées (p.65).

3.2 Descripteurs

3.2.1 BoW

Bag of Words ou Sac de Mots, est une technique permettant de décrire un document par rapport aux mots qu'il contient sous forme de dictionnaire. Suite à la formation de ce dictionnaire, les mots les moins présents sont enlevés car ce sont ceux qui portent le moins de sens. Finalement un histogramme est calculé par rapport au nombre de mots détectés. La comparaison de deux documents se fait donc en comparant leurs histogrammes respectifs. Il est également possible de définir une liste de rejet, contenant des mots à ne pas prendre en compte lors de la définition du dictionnaire (pronoms, articles...) car moins porteurs de sens et donc moins discriminants.

3.2.2 BoF

BoF ou *Bag of Features* aussi connu comme *BoVW* ou *Bag of Visual Words* est une technique permettant de décrire une image reposant sur ses aspects visuelles (points d'intérêt). La procédure est définie comme suit [8] :

— Extraction des points d'intérêt (utilisation de SIFT ou SURF)
— Partitionnement en k groupes représentant les k dimensions de l'histogramme qui représentera l'image
— Regroupement des points d'intérêt par rapport à leur groupe d'appartenance (utilisation de k-NN par exemple) et création de l'histogramme.

Les images sont ensuite comparées grâce à un algorithme d'apprentissage supervisé (SVM).

4 Méthodes de classification

4.1 k-PPV (K-NN)

k-NN (k-nearest neighbor) ou k-PPV (k plus proches voisins) est une méthode d'apprentissage supervisée. Cette méthode consiste à calculer la distance entre un point et les k points qui lui sont les plus proches. Pour calculer cette distance, on utilise la distance euclidienne :

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{d} (x_{ik} - x_{jk})^2} \tag{1}$$

Appliqué à la reconnaissance des documents, cette méthode va permettre de comparer les vecteurs de caractéristiques d'une image à classer, avec les vecteurs de caractéristiques des images de la base d'apprentissage. On choisi ensuite la classe majoritaire parmi ses k plus proches voisins. Pour déterminer k on peut également utiliser la technique de validation croisée.

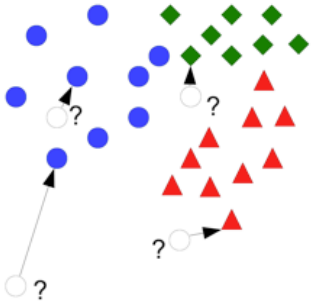


Figure 10 – *Exemple de classification par k-PPV*

4.2 SVM

SVM (séparateur à vaste marge ou *support vector machine* est, tout comme k-PPV, une méthode d'apprentissage supervisée. SVM est cependant plus complexe que cette dernière. SVM se base sur les notions de hyperplan, marges et vecteurs support. Pour un ensemble de données, SVM va trouver un classificateur linéaire qui permet de séparer les différentes classes, l'hyperplan. On détermine ensuite les points les plus proches de cet hyperplan, qui constitueront les vecteurs de support. Ces vecteurs de support permettront de calculer la marge maximale de l'hyperplan. La Figure 11 à été reprise de [5] et démontre l'application de SVM sur un ensemble de données.

Il peut exister une multitude de plans séparateurs (hyperplans) possibles pour un jeu de données. Pour résoudre ce problème, SVM cherche l'hyperplan optimale : celui qui maximise la marge définie par les vecteurs de support.

Lors que les données ne sont pas linéairement séparables, SVM permet de changer leur espace de représentation (espace de re-description). Plus cet espace est grand, plus il est probable de trouver un hyperplan séparateur convenable.

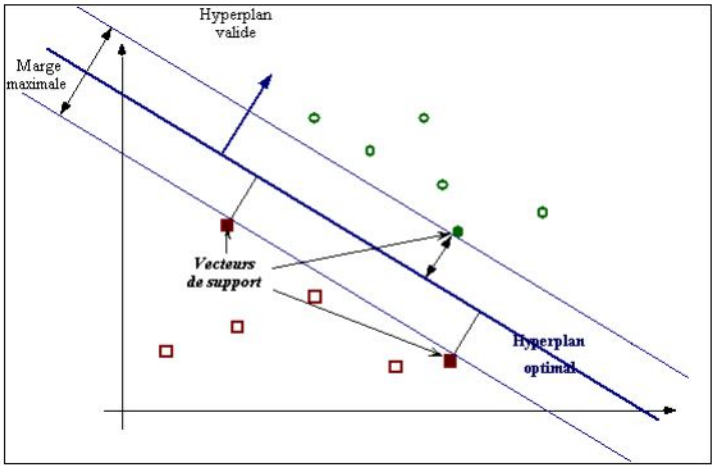


Figure 11 – *Exemple de classification par SVM*

4.3 One-class Classifier

Dans [12] et [13] les auteurs proposent une nouvelle méthode de classification basé sur une spécialisation des caractéristiques et la paramétrisation du classificateur en prenant compte de la classe du document à traiter. La méthode se divise en 3 étapes :

Extraction des caractéristiques

Pour l'extraction de caractéristiques, les auteurs se sont appuyés sur les caractéristiques visuelles et structurelles d'un document binarisé. Se basant sur des techniques existantes dans la littérature, la composition du vecteur se passe en trois parties :

- **Les caractéristiques de premier et arrière-plan :** ratio hauteur-largeur, densité de pixels noirs, centre de gravité de l'image et projections horizontales et verticales des pixels noirs
- **Les caractéristiques dites "primitives" :** nombre de composantes connexes, nombre d'occlusions, histogramme des lignes droites et histogramme des contours
- **Les caractéristiques par zone :** le document est divisé en 9 zones de taille identique et pour chaque zone il est calculé le nombre de composantes connexes, le nombre d'occlusions et le nombre de lignes

Spécialisation du vecteur de caractéristiques

Afin d'améliorer la qualité de la classification, les auteurs ont mis en place un système qui permet de spécialiser le vecteur de caractéristiques en supprimant des caractéristiques les moins pertinentes. Pour cela une estimation de la robustesse de chaque caractéristique est effectuée sur un ensemble de test de documents de même classe, permettant de définir un score de stabilité. Les N caractéristiques les plus robustes constitueront le vecteur définitif.

Classification

Les auteurs définissent le classificateur [1] comme démontré dans la Figure 12 avec x et t des vecteurs de caractéristiques, C une classe de document, M la distance euclidienne entre tous les éléments du set d'apprentissage, $d(x,t)$ la distance entre x et t, α le paramètre définissant la tolérance (taux de rejet) et K le nombre de voisins les plus proches à prendre en considération.

D'après les résultats fournis par les auteurs dans [12], représentés dans la Figure 13, ce classificateur fonctionne particulièrement bien pour des bases de petite taille (10 documents d'apprentissage seulement).

$$\gamma(x, C) = \begin{cases} 1 & if \quad \sum_{k=1}^{K} \delta(x, t_k) > \frac{K}{2} \\ O & otherwise \end{cases}$$

$$with\ \delta(x, t) = \begin{cases} 1 & if \quad \alpha d(x, t) > M \\ 0 & else \end{cases}$$

Figure 12 – *Formalisation du classificateur*

dataset	K-NN	SVM	OC K-NN
INS_ATT	78.00%	62.00%	**93.10%**
BANK_S	83.30%	83.33%	**100.00%**
BANK_B	86.40%	84.84%	**92.02%**
BANK_M	76.90%	56.41%	**94.18%**
TAXE_B	92.50%	88.67%	**93.53%**
TAXE_F	78.90%	78.40%	**79.74%**
FAM_BOOK	88.85%	88.46%	**95.68%**
CAR_REG	90.90%	63.63%	**98.92%**

Figure 13 – *Résultats obtenus avec des paramètres optimisées pour un meilleur taux de classification*

5 Librairies et *Frameworks*

5.1 Librairies

5.1.1 FLANN

FLANN [WWW9] ou *Fast Library for Approximate Nearest Neighbors* est une librairie d'algorithmes permettant d'effectuer des traitements de recherche de type "plus proche voisin". Cette librairie possède également un système permettant de choisir l'algorithme le plus adapté pour traiter un problème posé, choix basé sur l'ensemble des données à traiter.

Cette librairie est codée en C++ mais possède des *wrappers* permettant son utilisation avec d'autres langages de programmation.

Dans [2], Olivier Augereau utilise cette librairie pour effectuer une mise en correspondance rapide entre les points d'intérêt de l'image à traiter avec ceux de toutes les images de la base d'apprentissage.

5.2 Frameworks

5.2.1 Framework Aforge.NET

Aforge.Net [WWW17] est un *framework* développé en C# et contenant des algorithmes d'intelligence artificielle, traitement d'images, de *machine learning* ou encore des algorithmes génétiques.

5.2.2 Accord.NET

Accord.NET [WWW13] est un *framework* regroupant des algorithmes de *machine learning*, traitement d'images et traitement audio. Le *framework* est quasi-entièrement développé en C# et est utilisable pour des applications commerciales.

5.3 Logiciels d'OCR

Le projet traitant de la reconnaissance de documents et l'extraction de données d'intérêt, il est important d'avoir une idée précise des solutions d'OCR existantes sur le marché. Voici donc un descriptif des solutions intéressantes pour ce projet :

5.3.1 FineReader (Abbyy)

FineReader [WWW2] est un logiciel d'OCR développé par la société Abbyy. Ce logiciel reconnais plus de 190 langages (dans sa version 12) et intègre la technologie ADRT [1] [WWW1] qui crée un modèle logique basé sur la structure du document à analyser. Cela permet de détecter des zones identiques sur un ensemble de documents et de les traiter ensemble. Grâce à l'ensemble des technologies propriétaires, le logiciel à une précision de reconnaissance allant jusqu'à 99.8%. FineReader propose une interface graphique qui facilite la manipulation des données et est capable de fournir des résultats sous format *Word*, PDF ou *plain-text*. Cependant, ce logiciel est payant et n'est compatible qu'avec les systèmes d'exploitation Windows et Mac OSX.

5.3.2 Omnipage (Nuance Communications)

OmniPage [WWW24] est un logiciel d'OCR développé par la société Nuance Communications. Il est capable de reconnaître 123 langages différents et est compatible avec la plupart des scanners du marché. Omnipage intègre les outils SET [2] qui améliorent la qualité des documents scannées en terme de lisibilité et possèdent également la capacité de réduire les marges blanches. Ce logiciel est capable de fournir des résultats sous le format *Word*, *PowerPoint*, *HTML* ou bien PDF. Omnipage est cependant payant et n'est compatible qu'avec le système d'exploitation Windows.

5.3.3 Tesseract

Tesseract [WWW30] est un moteur d'OCR gratuit et open-source (Apache Licence v2.0). Ce moteur supporte plus de 100 langages (de base) et propose un système d'apprentissage permettant d'en supporter plus. Ce moteur est compatible avec un grand nombre de systèmes d'exploitation (dont Linux) [WWW31] et possède un grand nombre de *wrappers* permettant sa manipulation depuis plusieurs langages de programmation (dont .Net [WWW5]). Tesseract intègre également des algorithmes de pré-traitement [WWW29] tels le redimensionnement, la binarization ou la réduction du bruit. Le moteur est également capable de fournir des résultats sous le format *plain-text*, *hOCR* [WWW32] ou PDF.

Dans [2], Olivier Augereau présente une comparaison détaillée entre FineReader et Tesseract. Grâce à cette comparaison nous pouvons remarquer que ce moteur possède quelques

1. Technologie de Reconnaissance de Documents Adaptative
2. Scanner Enhancement Technology

limites tels la résolution de l'image (mauvaise lecture en dessous de 200 dpi), la robustesse à la rotation (mauvaise lecture au delà de 5°) ou incompatibilité avec l'écriture manuscrite. Exceptant l'écriture manuscrite, ces problèmes peuvent être résolus grâce au pré-traitement de l'image.

5.3.4 OCRopus

OCRopus [WWW33] est un logiciel d'OCR gratuit et open-source (Apache Licence v2.0), développé avec l'aide de Google pour leur projet *Google Books*. Ce logiciel est écrit en C++ et Python et a été développé pour les distributions Linux. OCRopus utilise le moteur *tesseract* et intègre un système d'analyse de structure et reconnaissance des caractères. Ce logiciel est surtout destiné au traitement de grandes quantités de documents. Nativement OCRopus s'utilise en ligne de commande et exporte ses résultats dans le format *hOCR* [WWW32].

6 Technologies

6.1 Docker

Docker [WWW19] est un logiciel libre (Apache Licence 2.0) de gestion de conteneurs, intégrant tout le nécessaire pour exécuter des programmes. Comme mentionné dans [WWW20], ces conteneurs permettent d'exécuter les applications toujours de la même manière, indépendamment de l'environnent dans lequel ils sont installés. A l'instar des VM [3], Docker partage le même *Kernel* avec l'ensemble des conteneurs, ce qui permet d'avoir des environnements d'exécution plus légers et donc plus facilement maintenables et partageables.

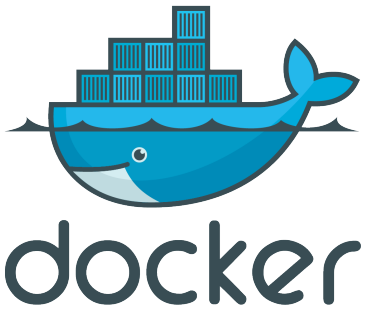


Figure 14 – *Logo de Docker*

6.2 AMQP et RabbitMQ

6.2.1 Protocole AMQP

L'AMQP [4] [WWW25] est un protocole d'échange qui permet la gestion de files de messages et qui à pour but de faire communiquer plusieurs applications de manière sûre en s'abstrayant de

3. Machines virtuelles
4. Advanced Message Queuing Protocol

leur implémentation. Comme l'auteur l'explique dans [WWW6], le protocole fonctionne par étapes.

Tout d'abord le client commence par effectuer une connexion avec le *broker*. Cette connexion permet l'ouverture de canaux qui vont ensuite être utilisés pour l'échange des messages. Suite à cette connexion, le client pourra accéder aux échangeurs et aux files d'attente. Un message émis par le client va ainsi être déposé sur un échangeur qui se chargera de faire suivre le message (grâce aux *bindings* [5]) vers la file correspondante. Une fois stockée dans la file, un message sera consommé par une application. Cette application va envoyer une requête au *broker* qui lui transmettra le prochain message de la file (et le supprimera de la file par la même occasion). Un *broker* peut ainsi contenir plusieurs échangeurs et plusieurs files d'attente. Chaque échangeur peur être lié à plusieurs files d'attente, ce qui permet la transmission d'un même message à plusieurs files en même temps.

Dans la Figure 15, extraite de [WWW6], nous pouvons voir la structure interne d'un échange par le protocole AMQP :

— **P :** représente un producteur de messages
— **X :** représente un échangeur
— **C :** représente un consommateur de messages

Nous remarquons qu'un producteur est lié à un échangeur qui transmet les messages sur les deux files aux quelles il est lié. Chaque consommateur est ensuite lié à une file et ne *consomme* que les messages qui y sont stockés.

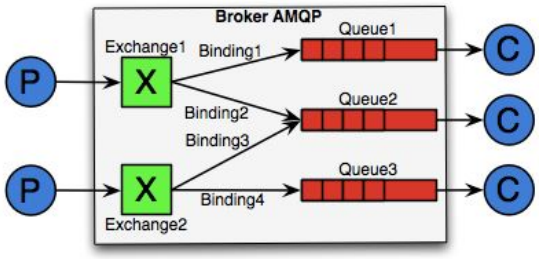


Figure 15 – *Diagramme AMQP*

6.2.2 RabbitMQ

RabbitMQ est une implémentation du protocole AMQP par Pivotal [WWW26]. Dans [WWW21], l'auteur nous explique que RabbitMQ est notamment utile pour l'exécution de tâches distribuées. Ainsi, grâce à son système de files d'attente, le processus producteur peut même être dans un serveur différent de celui du processus consommateur. Ce partage des tâches soulage ainsi le serveur principal (par exemple dans le cas d'un site internet) qui n'a plus besoin d'effectuer la tâche demandée dés sa réception. La tâche est mise dans une file d'attente, et un autre processus viendra s'en charger dés qu'il sera libre. Cela permet également un très faible couplage entre producteur et consommateur. La modification de l'un n'affecte pas l'autre, et la remise en état du système se fera beaucoup plus aisément. RabbitMQ propose également un système de *plugins* permettant d'ajouter de nouvelles fonctionnalités au système et apporte une couche sécurité aux bases du protocole AMQP. Il permet ainsi une mise en place d'un cluster plus facilement, ce qui apportera une meilleur tolérance aux pannes et aux montées en charge [WWW7].

5. Liaison

RabbitMQ est développé en Erlang [WWW7], langage développé par Ericsson et destiné aux applications distribuées [WWW8].

Figure 16 – *Logo de RabbitMQ*

4 Analyse et Conception

1 Étude de faisabilité : Mono

Mono est une plateforme logicielle open-source qui implémente le *framework* .Net de *Microsoft*. De part son caractère *cross-platform*, et dû aux contraintes de développement du projet (environnement Linux), Mono est la solution idéale permettant d'allier les nombreuses bibliothéques de traitement existantes en C# (comme Aforge ou Accord) et l'environnement d'exécution demandé.

Cependant, il est nécessaire de connaître la stabilité de cette plateforme sous l'environnement Linux avant de prendre des décisions concernant les bibliothèques et algorithmes à utiliser durant le projet et donc, avant la phase de développement. Pour cela une étude de faisabilité à été effectué concernant Mono.

Tout d'abord une VM Linux à été crée correspondant au système du client, c'est à dire exécutant une distribution Linux Debian 8 (jessie). Puis, en suivant les consignes du site de l'éditeur [WWW27], l'installation de la plateforme à été effectuée. Le premier constat est que la procédure expliquée sur [WWW27] n'est pas compléte. En effet il manque des dépendances qui rendent l'installation de certains composantes impossible. Suite à quelques recherches, il a été possible de définir une procédure d'installation complète et fonctionnelle, décrite en annexe de ce document.

Suite à l'installation de la plateforme Mono, l'installation de l'IDE MonoDevelop à été effectuée. Ici de même, la procédure n'était pas complète. Quelques recherches ont également permis de surmonter l'obstacle.

Une fois l'IDE installé, des tests ont été effectués. Des programmes simples ont été exécutés en mode console et se sont bien déroulés.

Suite à ces tests nous pouvons considérer l'environnement stable et apte à être utilisé lors de ce projet.

2 Tests de stabilité des librairies *Aforge.net* et *Accord.net* sous Mono

L'objectif de ce tests est de vérifier la stabilité des librairies *Aforge.net* et *Accord.net* sous le système d'exploitation Linux et l'environnement d'exécution Mono et ainsi étoffer l'étude précédente [1]. Cela permettra ainsi de confirmer l'utilisation de ces technologies (Mono, MonoDevelop) pour la suite du projet.

Pour la réalisation de ces tests, nous avons utilisé des codes source proposés dans les documentations respectives des librairies à tester. Tout d'abord car il s'agit seulement de tests (donc pas de code de production) puis car cela nous permettra de comparer les résultats obtenus par l'éditeur avec les résultats obtenus sur notre environnement d'exécution.

2.1 Tests sur *Aforge.net*

Classe *CloudsTexture*

Tout d'abord une création de textures avec la classe *CloudsTexture*, documentée sur [WWW15]. Ce test permet également de tester l'écriture de données sur le disque (enregistrement de l'image obtenue).

Nous avons donc repris le code proposé sur la page (légèrement) modifié pour les besoins du test :

```
1 // create texture generator
2 CloudsTexture cText = new CloudsTexture ();
3 // generate new texture
4 float[,] texture = cText.Generate (320, 240);
5 // convert it to image to visualize
6 Bitmap textureImage = TextureTools.ToBitmap (texture);
7 // save result to disk
8 textureImage.Save ("testAforge.jpg", System.Drawing.Imaging.ImageFormat.Jpeg);
```

Le résultat obtenu peut être visualisé dans la Figure 1 et s'approche du résultat obtenu par l'éditeur.

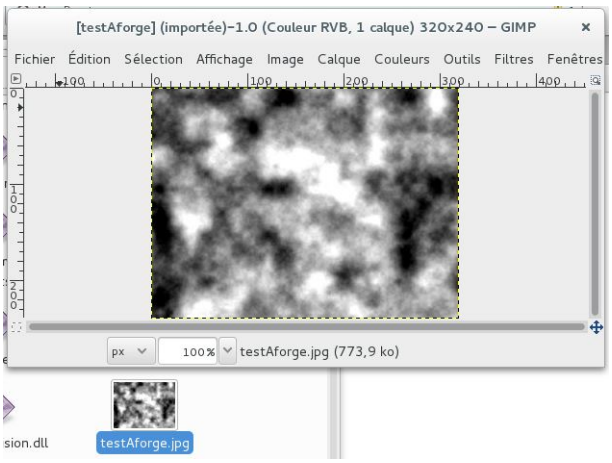


Figure 1 – *Texture générée par CloudTexture*

1. Étude de faisabilité : Mono

Classe *Convolution*

Pour le second test, un traitement plus complexe à été choisi : un filtre de convolution appliqué à une image de référence en utilisant la classe *Convolution* documentée sur [WWW16]. Ce test permet également de tester le chargement de données à partir du disque.

Comme pour le test précédent, nous avons donc repris le code proposé sur la page (légèrement) modifié :

```
1  // load sample from disk
2  Bitmap sample = new Bitmap("sample5.jpg");
3  // define emboss kernel
4  int[,] kernel = {
5      { -2, -1,  0 },
6      { -1,  1,  1 },
7      {  0,  1,  2 } };
8  // create filter
9  Convolution filter = new Convolution( kernel );
10 // apply the filter
11 Bitmap result = filter.Apply (sample);
12 // save result to disk
13 result.Save ("ImagingTest.jpg", System.Drawing.Imaging.ImageFormat.Jpeg);
```

La Figure 2 nous montre le résultat obtenu (à droite) pour l'image de référence donnée (à gauche). Nous pouvons également remarquer que le résultat coïncide avec celui obtenu par l'éditeur.

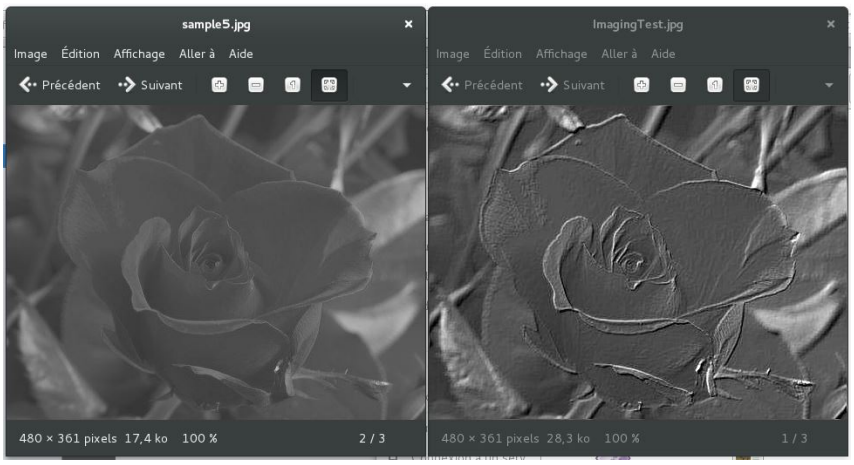


Figure 2 – *Résultat de convolution*

Conclusion

Suite à ces deux tests, nous pouvons conclure que la librairie est fonctionnelle dans l'environnement d'exécution Mono.

2.2 Tests sur *Accord.net*

Pour les tests sur la librairie *Accord.net* nous avons suivi le même principe que pour *Aforge*. Le code utilisé est référencé dans [WWW14] et utilise SVM :

```
double[][] inputs = {
  /* 1.*/ new double[] { 0, 0 },
  /* 2.*/ new double[] { 1, 0 },
  /* 3.*/ new double[] { 0, 1 },
  /* 4.*/ new double[] { 1, 1 },
};

int[] outputs = {
  /* 1. 0 xor 0 = 0: */ -1,
  /* 2. 1 xor 0 = 1: */ +1,
  /* 3. 0 xor 1 = 1: */ +1,
  /* 4. 1 xor 1 = 0: */ -1,
};

// Create a new machine with a polynomial kernel and two inputs
var ksvm = new KernelSupportVectorMachine(new Gaussian(), 2);

// Create the learning algorithm with the given inputs and outputs
var smo = new SequentialMinimalOptimization(ksvm, inputs, outputs){
  Complexity = 100 // Create a hard-margin SVM
};

// Teach the machine
double error = smo.Run();
Console.WriteLine("error:" + error);

// Show results on screen
ScatterplotBox.Show("Training data", inputs, outputs);
ScatterplotBox.Show("SVM results", inputs, inputs.Apply(p => ↩
        System.Math.Sign(ksvm.Compute(p))));

Console.ReadKey();
```

Comme nous pouvons le remarquer dans la Figure 3, les résultats obtenus coïncident exactement avec ceux obtenus par l'éditeur.

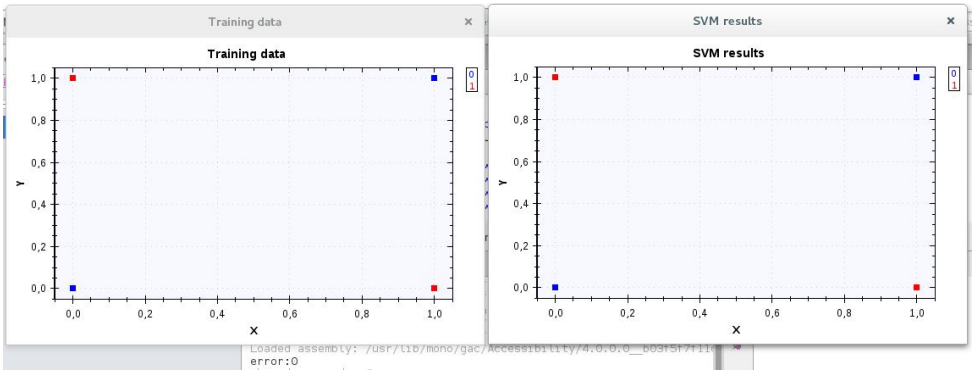


Figure 3 – *Résultat de SVM*

Conclusion

Suite à ce test, nous pouvons également conclure que la librairie est fonctionnelle dans l'environnement d'exécution Mono.

Deuxième partie

Cahier de Développement

5 Mise en Œuvre

1 Méthodologie

Lors de la phase d'analyse, un premier diagramme de classes à été réfléchis et rédigé. Cela à posé une base au projet et à permis d'établir un fil conducteur. Au fur et à mesure du développement, des modification ont été apportées à ce diagramme afin de mieux répondre aux objectifs du projet. Un système de *versionning* à également été mis en place avec *Git* et *GitLab* ainsi que le logiciel *GitKraken*, qui à amélioré la gestion du *repository*.

Ce projet à été développé avec Visual Studio 2015 (version utilisé consultable sur la Figure 1) et le *framework* .Net dans sa version **4.5.2**, afin de garder une compatibilité maximale avec Mono. Concernant les conventions de nommage, un effort à été effectué afin de respecter au maximum les convention établies par Microsoft ([WWW23]).

Au départ l'objectif était d'utiliser la plateforme Mono sous Debian afin de créer un environnement de développement proche de celui de production. Du à la faible puissance de la machine de développement, la machine virtuelle *Debian* crée pour l'occasion à engendré des soucis de performance. Afin d'être plus productif et avec l'accord des encadrants et du client, il à été décidé de basculer l'environnement de développement sous *Windows*.

Certains *Design Patterns* ont également été mis en place, notamment des *interfaces* et des *factory*. Les *interfaces* permettent une évolution et maintenance aisée des algorithmes. Par exemple, actuellement seul le moteur Ocr Tesseract est implémenté. La mise en place d'une *interface* permettra donc l'utilisation d'un autre moteur Ocr sans devoir rechanger d'autres parties du projet. Le second *Design Pattern* utilisé, les *factory*, permettent la création d'instances de classe sans utiliser le mot-clé *new*. Cela améliore ainsi la lisibilité du code et la maintenance de ce dernier car si la méthodologie de création d'une classe change, seulement quelques petites modifications ne seront à apporter au projet.

Un effort supplémentaire à également été apporté afin de respecter certaines consignes décrites dans [7].

Concernant la documentation, celle ci à été rédigée en respectant les conventions Microsoft [WWW22]. L'outil *DOxygen* à été utilisé pour leur extraction et formatage *HTML* / *PDF*.

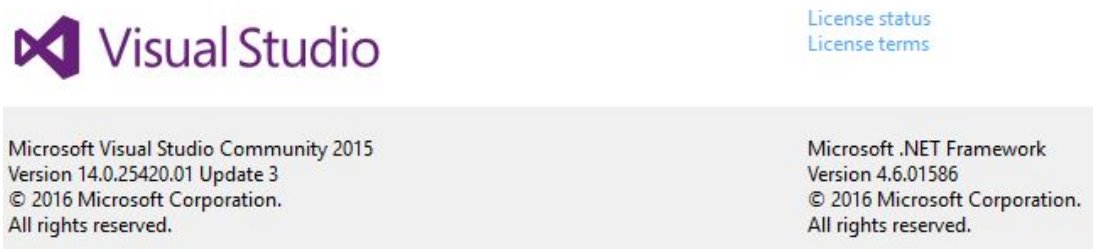


Figure 1 – *Version de Visual Studio utilisée*

2 Dépendances

Plusieurs librairies externes ont été utilisées lors du développement de ce projet. A part les librairies fournies par l'équipe RFAI, les autres dépendances sont gérées directement par le gestionnaire de *packages* ***NuGet***.

2.1 Libraires de l'équipe RFAI

Deux librairies de l'équipe RFAI ont été utilisées :

- **FeaturesExtractors**
  - **Version : 1.0.0.0 (07/07/2015)**
  - Cette librairie propose des algorithmes et fonctionnalités d'extraction de caractéristiques.
- **IntrinsicClassification :**
  - **Version : 1.0.0.0 (02/03/2017 recompilée)**
  - Cette librairie propose des algorithmes de classification (dont le *OneClass Knn* mentionné dans l'état de l'art).

2.2 Autres dépendances

Quatre librairies externes ont été également utilisées pour le développement de ce projet. Ces librairies sont toutes libres d'utilisation :

- **Accord.Net Framework**
  - **Core**
    - **Version : 3.3.0 (16/09/2016)**
    - Fonctionnalités de base du *framework* Accord. Indispensable pour l'utilisation des autres fonctionnalités du *framework*.
  - **Imaging**
    - **Version : 3.3.0 (16/09/2016)**
    - Propose des fonctionnalités de gestion et traitement d'images. Très utile dans le contexte de ce projet notamment concernant les pré-traitements à appliquer au document (binarization, filtres...).
  - **Math**
    - **Version : 3.3.0 (16/09/2016)**
    - Propose des algorithmes et fonctionnalités de résolution mathématique.
- **HtmlAgilityPack :**
  - **Version : 1.4.9.4 (29/06/2016)**

  - Cette librairie propose des fonctionnalités de gestion de données sous format HTML. Hocr étant basé sur la structure HTML, cette librairie nous permet de mieux gérer les données extraites par le moteur Ocr.
- **log4net :**
  - **Version : 2.0.7 (05/01/2017)**
  - Cette librairie permet une gestion des Log. Le système étant prévu (dans le futur) pour être exécuté de manière autonome, il est important d'avoir un retour sur sont fonctionnement. La meilleur solution est donc de mettre en place un système de log.
- **Tesseract :**
  - **Version : 3.0.1 (23/12/2015)**
  - Il s'agit d'un *wrapper* .NET de **tesseract-ocr 3.04** [WWW5].

3 Métriques

Afin de maintenir une qualité de code constante, un serveur *Sonar* à été mis en place et lié directement à Visual Studio grâce au plugin *SonarLint*. Cela à permis d'avoir un retour en direct sur la qualité du code fourni, et donc de prévoir d'éventuelles modification, corrections ou *refactoring*. Des analyses hebdomadaires ont également été effectuées afin d'obtenir des bilans plus détaillés. La Figure 2 démontre les métriques correspondantes aux duplication de code.

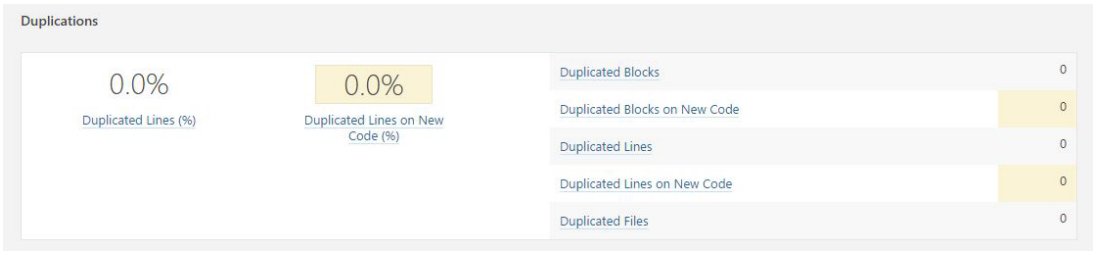


Figure 2 – *Rapport Sonar : Duplications de code*

La Figure 4 nous montre la fiabilité et le niveau de sécurité du code fournis et la Figure 3 les métriques correspondantes au niveau de maintenance et la couverture du code. Quant à la Figure 5, celle-ci démontre le détail du projet concernant les proportions de code rédigé et, enfin, la Figure 6 montre la complexité totale du projet ainsi que les problèmes rencontrés.

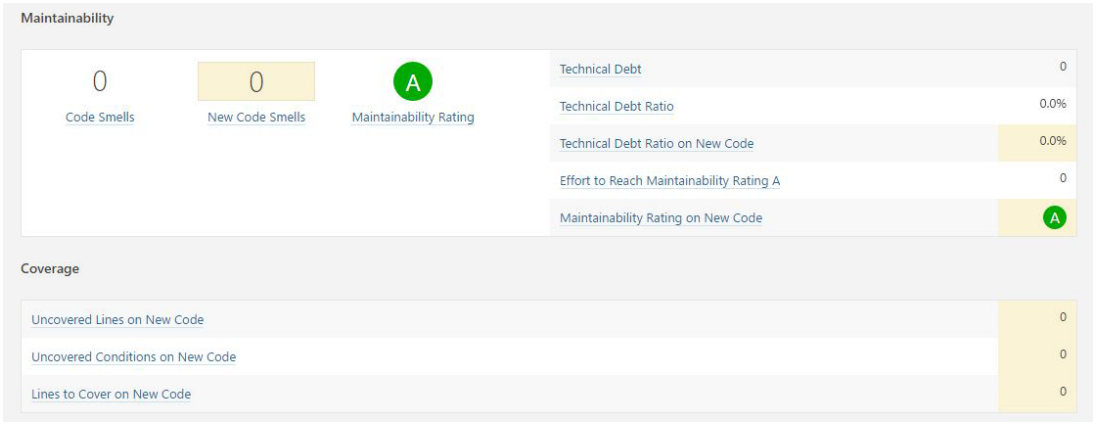


Figure 3 – *Rapport Sonar : Maintenance et Couverture*

Figure 4 – *Rapport Sonar : Fiabilité et Sécurité*

Figure 5 – *Rapport Sonar : Taille du projet*

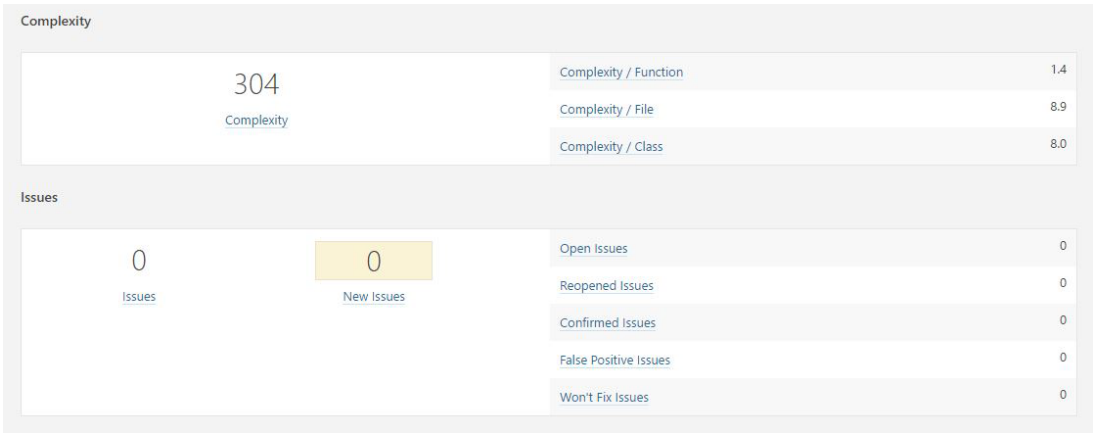


Figure 6 – *Rapport Sonar : Complexité et Problèmes*

6 Diagrammes de classes

Le système final est composé de 8 modules ainsi qu'une classe d'entrée. Ces composantes seront détaillées dans la suite du chapitre. Une présentation de la structure globale du système est également présentée en fin de chapitre.

1 Classe *ProcessDocument*

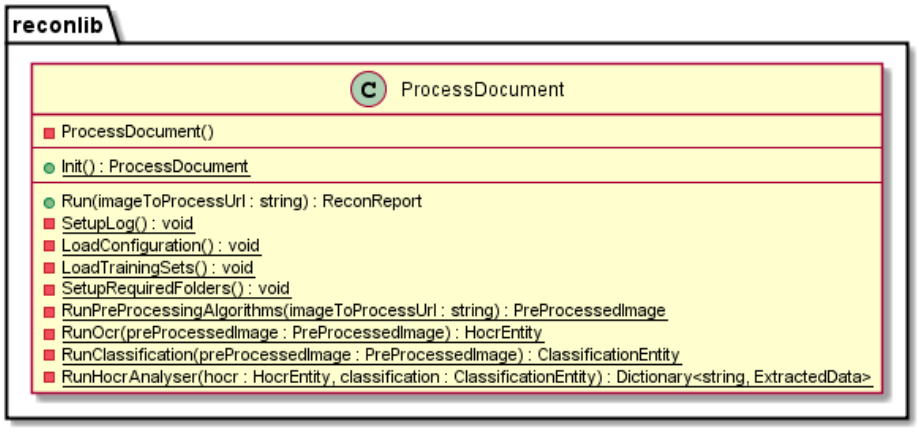


Figure 1 – *Diagramme de classe : ProcessDocument*

ProcessDocument est le point d'entrée du système. Cette classe, représentée dans la Figure 1, permet l'initialisation et l'exécution des services (reconnaissance, classification, extraction de données). Une fois initialisée (fichiers de configuration), l'utilisation de la méthode *Run* permet l'exécution de l'ensemble des services sur un document donné et fourni un rapport détaillé du processus.

2 Module *Analysis*

Le module *Analysis*, représenté par la Figure 2, fourni des services et modèles spécifiques à l'analyse de données. Son sous-module *Sections*, Figure 3, renseigne l'ensemble des modèles nécessaires au stockage de données sous format *hocr* et le sous-module *Hocr* des services d'analyse et extraction. Deux autres classes sont également proposées. *Levenshtein* permet la calcul de la distance entre deux mots, utile pour la détection de mots-clé, et la classe *MRZ* propose l'ensemble des fonctionnalités nécessaires pour traiter des *Machine Readable Zone* ou *MRZ* (détection, extraction, calcul d'intégrité).

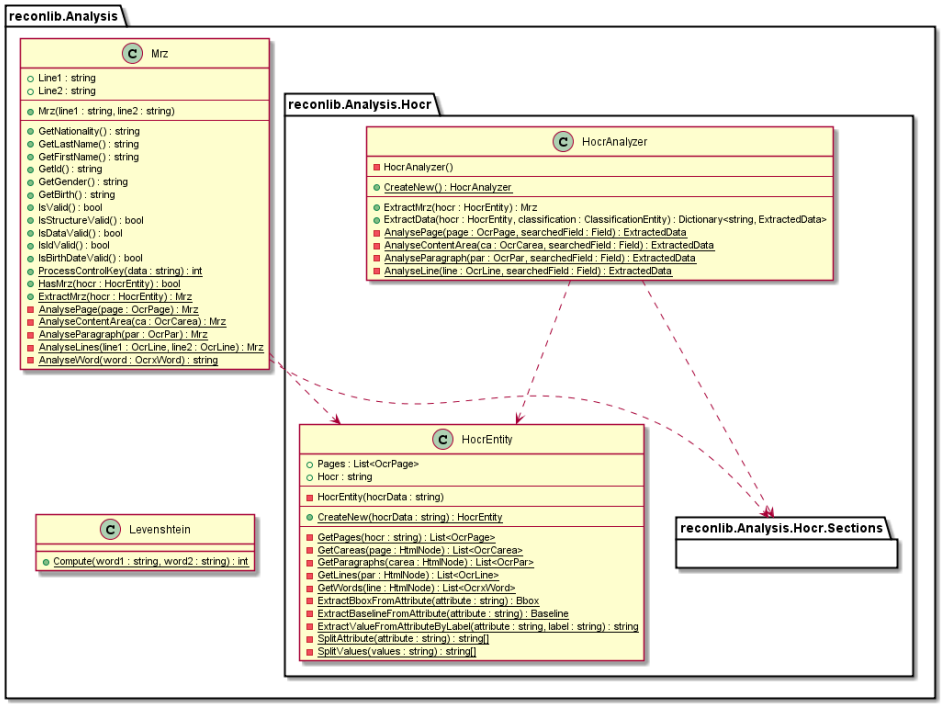


Figure 2 – *Diagramme de classe : Module Analysis*

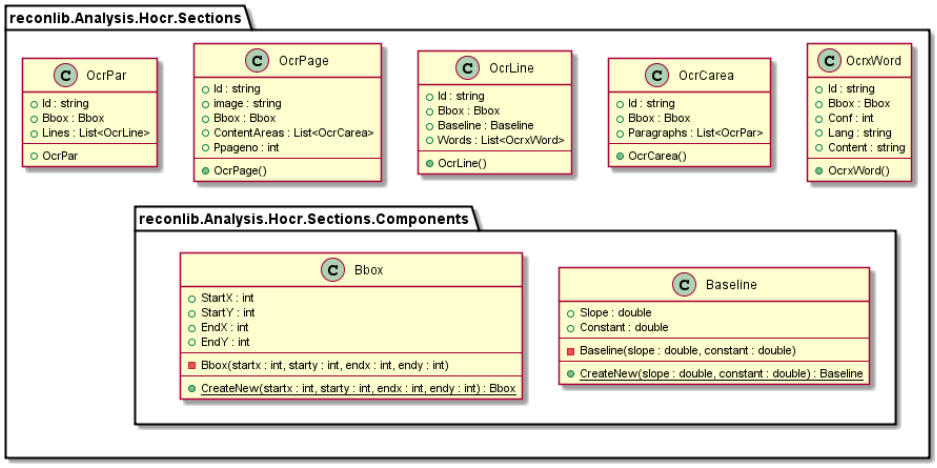


Figure 3 – *Diagramme de classe : Sous-Module Sections*

3 Module *Classification*

Le module *Classification*, représenté par la Figure 4, propose des services de classification de données. Actuellement seul un classificateur est disponible (*One-Class KNN*) basé sur les travaux de l'équipe RFAI. L'exécution du classificateur se fait à partir d'une méthode unique (*Run*). Une des évolution possibles (et recommandées) de ce module est de faire l'implémentation d'une interface. Cette interface proposerais notamment la définition de la méthode *Run* ce qui améliorerait grandement l'évolution et la maintenance du système, en permettant l'intégration aisée de nouveaux classificateurs.

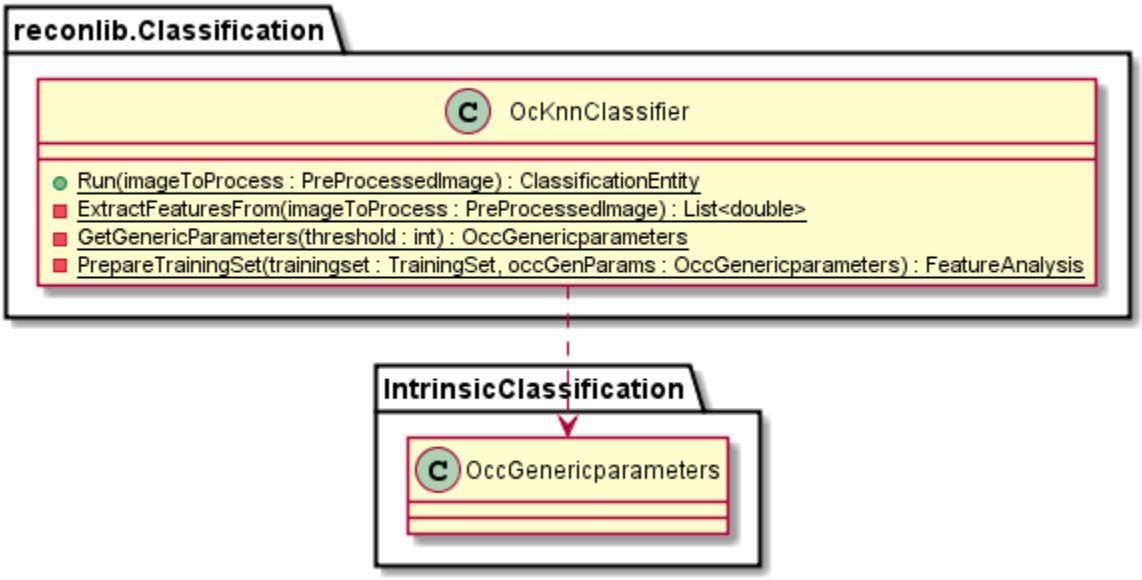


Figure 4 – *Diagramme de classe : Module Classification*

4 Module *Exceptions*

Le module *Exceptions*, représenté par la Figure 5, contient des implémentations d'exceptions non proposées par le *framework* .Net. Actuellement une seule exception est implémentée, *EmptyFolderException* qui permet le renvoi d'exceptions lors de la rencontre d'un dossier vise. Cette exception est notamment utile lors du chargement et traitement de la base d'apprentissage, où un dossier vide peut engendrer un mauvais fonctionnement du système.

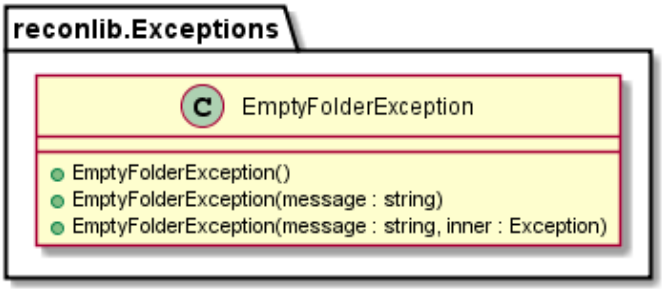


Figure 5 – *Diagramme de classe : Module Exceptions*

5 Module *FeatureExtraction*

Le module *FeatureExtraction*, représenté par la Figure 6, encadre les services d'extraction de caractéristiques. Cette fonctionnalité est basée sur une interface unique, permettant de rendre les algorithmes d'extraction implémentés utilisables dans le système, et améliorant l'évolution du module. Ces algorithmes sont donc déclarés et implémentés dans le sous-module *Algorithms* et doivent implémenter l'interface (et donc la méthode *Run*). Actuellement un seul algorithme est implémenté, *GlobalFeatures*, basé sur les travaux de l'équipe RFAI.

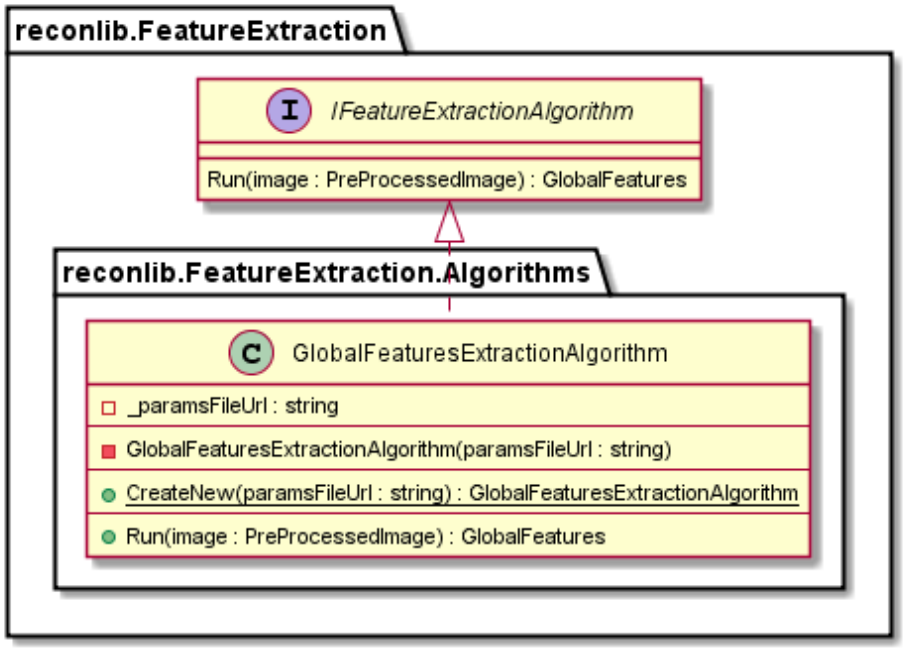


Figure 6 – *Diagramme de classe : Module FeatureExtraction*

6 Module *Models*

Le module *Models*, représenté par la Figure 7, encadre les modèles de données utilisés par le système. On compte 7 modèles :

- **PreProcessedImage :** Ce modèle est une des clés du système. Il est utilisé par la majorité des fonctionnalités car il enregistre l'image du document à traiter et effectue l'ensemble des pré-traitement définis sur l'image (binarisation et autres filtres).
- **ClassificationEntity :** Ce modèle représente le résultat de la classification. Il renseigne ainsi le résultat de la classification et le score obtenu lors de celle-ci.
- **TrainingSet :** Ce modèle enregistre l'ensemble des bases d'apprentissage. Instancié lors de l'initialisation du système, ce modèle permet d'avoir l'ensemble des données relatives à ces bases d'apprentissage (id, chemin et pattern d'analyse sémantique associé) chargées en mémoire et utilisables par les fonctionnalités concernées.
- **Configuration :** Ce modèle enregistre l'ensemble des données de configuration. Tout comme le modèle *TrainingSet*, les données de configuration sont chargées au démarrage du système et enregistrées dans une instance de *Configuration*, ce qui permet d'avoir ces données de configuration chargées en mémoire et utilisables par les fonctionnalités concernées.

- **ReconReport :** Ce modèle représente le rapport d'exécution du système sur un document. Il permet ainsi d'enregistrer le document analysé, la classification obtenue, les données extraites et le résultat de l'analyse sémantique.
- **ExtractedData :** Ce modèle représente une donnée extraite lors de l'analyse sémantique. Il enregistre ainsi la donnée extraite (1 ou plusieurs mots, dépendant du pattern associé) ainsi que la confidence d'extraction moyenne de cette donnée.
- **AnalysisPattern :** Ce modèle représente un pattern d'analyse sémantique. Ces patterns sont enregistrés sous format .xml et définissent ainsi la structure du document à analyser et vont permettre l'extraction de données ciblées sur ce dernier. Chaque base d'apprentissage est associée à un pattern d'analyse. Lors du chargement des bases d'apprentissage (cf. *TrainingSet*), les patterns associés sont également chargés. Le pattern est ensuite réutilisé après la phase de classification, lors de l'analyse et extraction sémantique.

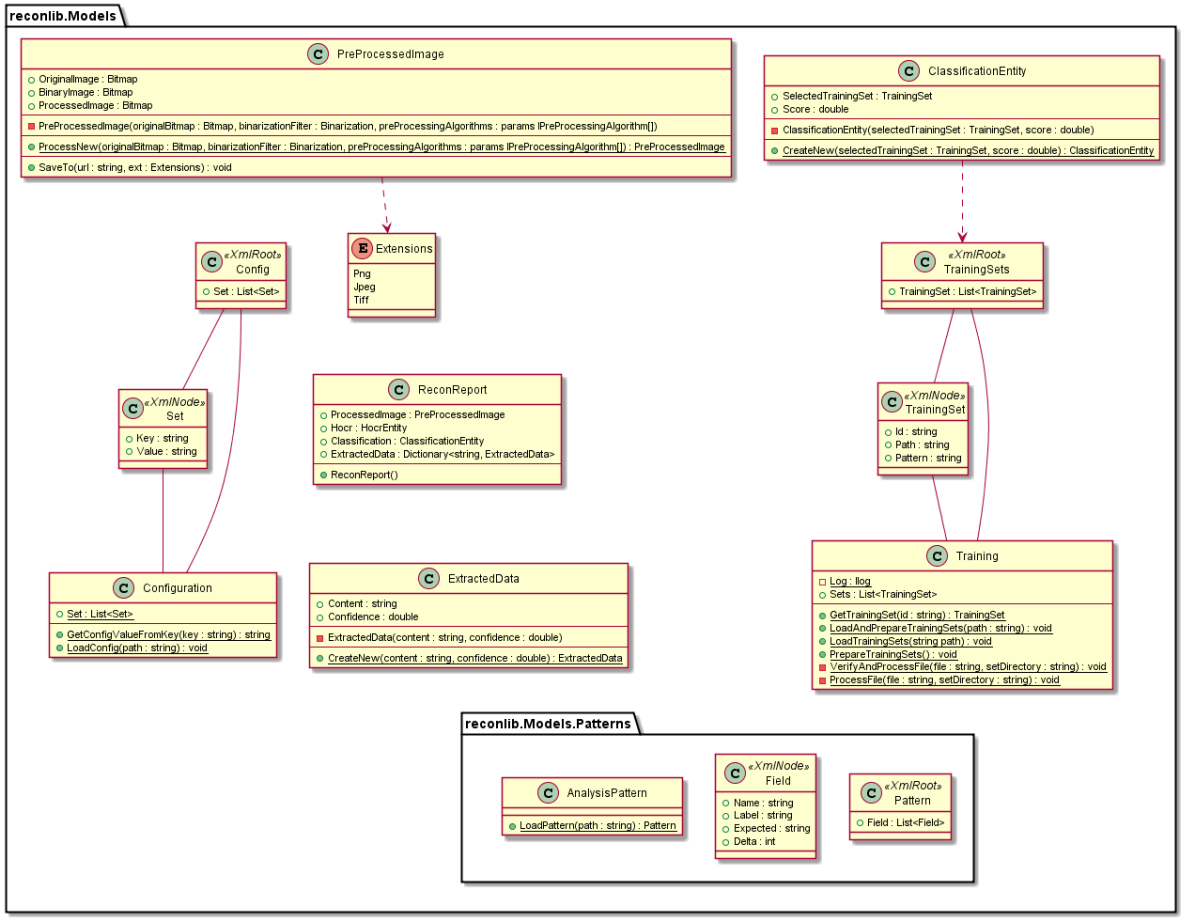


Figure 7 – *Diagramme de classe : Module Models*

7 Module *Ocr*

Le module *Ocr*, représenté par la Figure 8, encadre les services d'analyse Ocr. Tout comme les autres fonctionnalités d'intégration potentiel de libraires externes, cette fonctionnalité est basée sur une interface unique permettant l'utilisation de différents moteurs Ocr au sein du service, et améliore par la même occasion l'évolution et maintenance du module. Actuellement seul un *wrapper* du moteur *Tesseract* est implémenté.

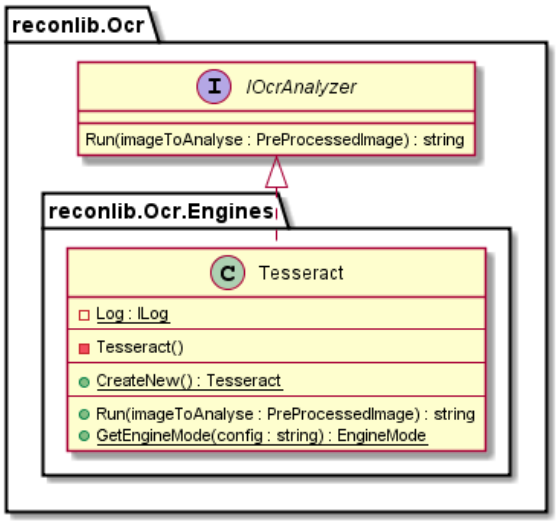


Figure 8 – *Diagramme de classe : Module Ocr*

8 Module *PreProcessing*

Le module *PreProcessing*, représenté par la Figure 9, propose des services de pré-traitement d'image. Basés sur une interface unique, le module permet une implémentation et utilisation aisées de nouveaux algorithmes et améliore l'évolution et maintenance du module par la même occasion. Actuellement quatre *wrappers* sont implémentés utilisant des algorithmes proposés par la librairie *Accord.Net* :

- **SkewCorrector :** Cet algorithme permet de redresser une image (voir *état de l'art*) et améliore par la même occasion les taux de reconnaissance des moteurs Ocr.
- **MedianFilter :** Ce filtre permet de lisser une image tout en gardant les formes de celle-ci. Une réduction du bruit d'une image améliore également le taux de reconnaissance des moteurs Ocr.
- **Binarization :** Cet algorithme transforme une image en image binaire (noir et blanc). L'application de ce filtre est primordiale lors d'une analyse Ocr (voir *état de l'art*).
- **AdaptativeSmooth :** Ce filtre est un replaçant au filtre Médian.

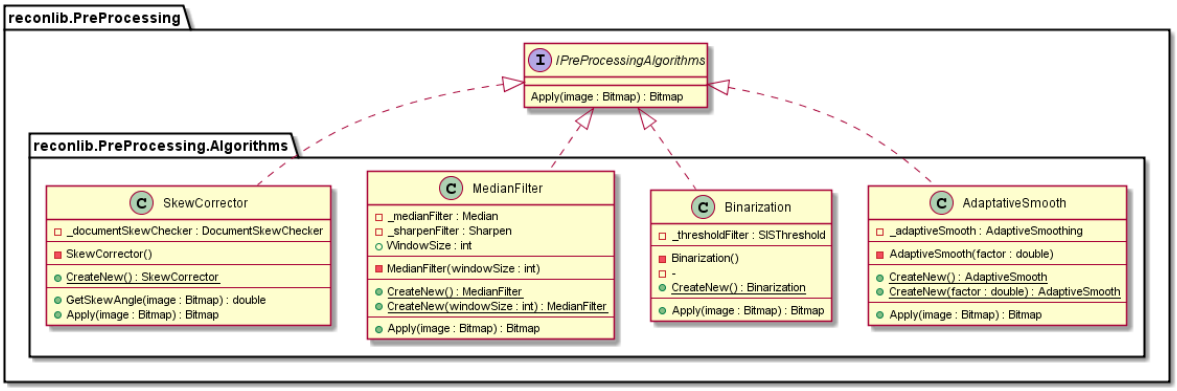


Figure 9 – *Diagramme de classe : Module PreProcessing*

9 Module *Tools*

Le module *Tools*, représenté par la Figure 10, encadre des classes *outils* utiles au système. Actuellement un seul outil est implémenté, *ImageTools*, qui permet de charger des images et de vérifier leur compatibilité avec le système.

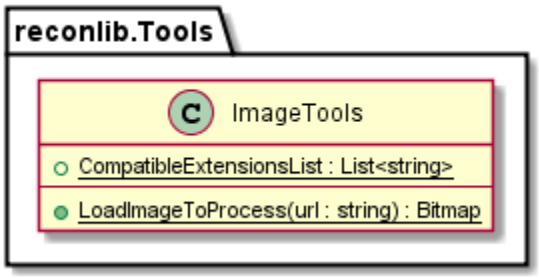


Figure 10 – *Diagramme de classe : Module Tools*

10 Structure globale du système

Due à sa grande taille, veuillez trouver le diagramme de la structure globale du système dans les annexes de ce rapport.

7 Campagne de Tests

1 Tests d'intégration

Par soucis de temps et charge de travail lors du développement, l'utilisation de librairies externes est indispensable. Néanmoins il est important de vérifier leur bonne intégration et fonctionnement au sein du projet.

1.1 Accord.Net

Lors du développement de ce projet nous avons utilisé la libraire *Imaging* du *framework* Accord afin d'appliquer des algorithmes de pré-traitement aux images manipulées.

Binarization

La documentation Accord [WWW12] nous renseigne que lorsque l'algorithme de binarization **SISThreshold** est appliqué à l'image de gauche Figure 1, le résultat attendu est celui de l'image de droite.

Figure 1 – *Test d'intégration Accord : Algorithme SISThreshold - Résultat espéré*

La Figure 2 représente le résultat obtenu après traitement de l'image de test par le système développé.

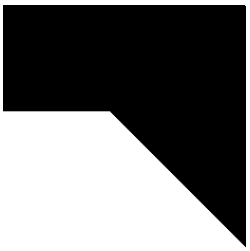


Figure 2 – *Test d'intégration Accord : Algorithme SISThreshold - Résultat obtenu*

Filtre médian

De même que pour le filtre *SISThreshold*, la documentation Accord [WWW11] nous renseigne que lorsque le filtre médian est appliqué sur l'image de gauche Figure 3, le résultat attendu est celui de l'image de droite.

Figure 3 – *Test d'intégration Accord : Filtre médian - Résultat espéré*

La Figure 4 représente le résultat obtenu après traitement de l'image de test par le système développé.

Figure 4 – *Test d'intégration Accord : Filtre médian - Résultat obtenu*

Document Skew Checker

La documentation Accord [WWW10] nous renseigne également que lorsque l'algorithme de *document skew checker* est appliqué à l'image de gauche Figure 5, le résultat attendu est celui de l'image de droite.

La Figure 6 représente le résultat obtenu après traitement de l'image de test par le système développé.

Conclusion

Suite aux résultats obtenus, nous pouvons conclure que le *framework* Accord est opérationnel et bien intégré au système en développement.

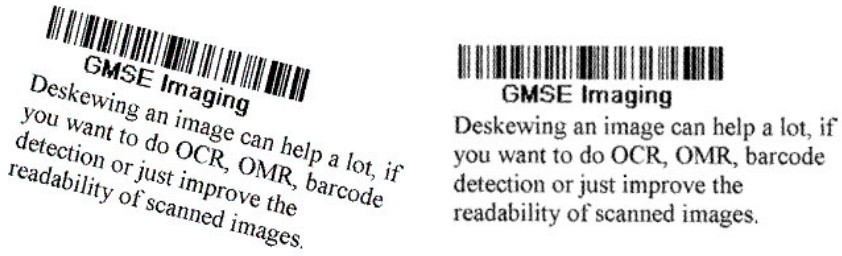


Figure 5 – *Test d'intégration Accord : Algorithme Document Skew Checker - Résultat espéré*

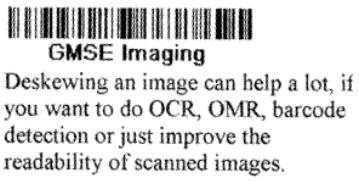


Figure 6 – *Test d'intégration Accord : Algorithme Document Skew Checker - Résultat obtenu*

1.2 log4net

La librairie *log4net* permet le monitoring le fonctionnement de l'application par moyen des log. Ces logs peuvent être affichées sur une console de sortie mais également écrits dans des fichiers crées pour l'occasion. Le système développé visant à être autonome, il est important que ces logs puissent être écrits dans des fichiers.

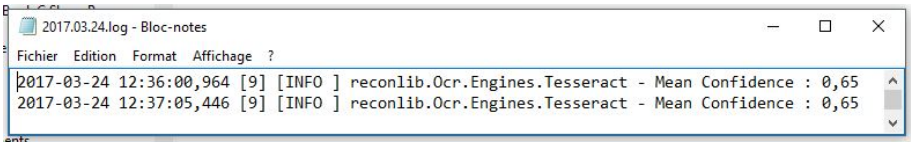


Figure 7 – *Test d'intégration log4net : Résultat obtenu*

La Figure 7 montre un des fichiers crées par l'application avec des informations d'exécution. Nous pouvons ainsi conclure que la librairie *log4net* est bien fonctionnelle au sein du système.

1.3 Tesseract

Afin de tester l'intégration du moteur Tesseract, un fichier exemple à été fourni et son contenu comparé à celui extrait par le moteur. D'après les résultats visibles sur la (Figure 8, nous pouvons conclure le bon fonctionnement du moteur ocr Tesseract au sein du système.

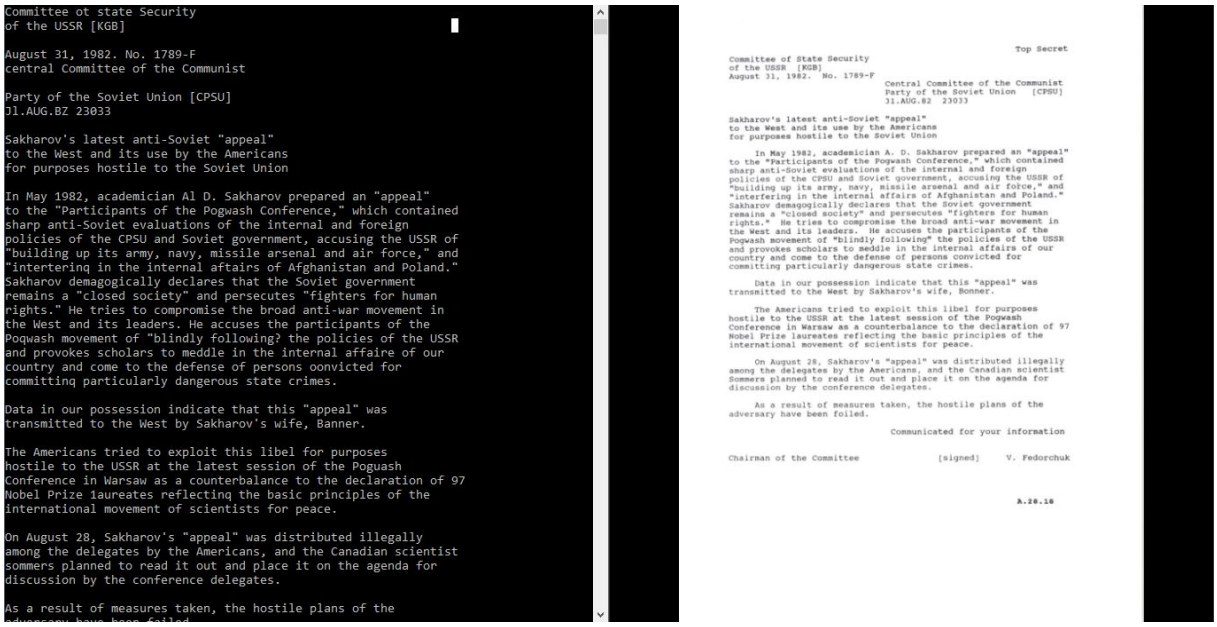


Figure 8 – *Test d'intégration Tesseract : Résultat obtenu*

2 Tests unitaires

Pour les testes unitaires, **NUnit 3** à été utilisé, relié à un système de tests en continu enclenchés par l'action de *Build*. Cela à permis de suivre l'état du projet en direct et pouvoir apporter des corrections si besoin. De plus, ce système est particulièrement pratique lors des phases de *refactoring*. En effet, le *refactoring* est une étape importante de la phase de développement car permet de nettoyer et restructurer le code et diminuer la complexité cyclique de ce dernier. C'est également une phase très délicate car lorsque une fonction est divisée par soucis de complexité, il peut arriver que son fonctionnement en soit affecté. Les tests en continu permettent de soulever cette erreur de fonctionnement en temps réel.

Due aux contraintes de temps du projet, il n'a pas été possible d'obtenir une couverture des tests unitaires de 100%. La Figure 9 démontre le rapport de tests unitaires ainsi que le taux de couverture globale et par fonctionnalité.

Un taux de couverture de 70% (global) à ainsi été obtenu, dont 85% de tests implémentés et 62% de tests positifs.

La Figure 10 montre les *points chauds* du système. La taille du nom de la classe est proportionnel au risque de *bug* que cette même classe peut engendrer. Actuellement la classe *HocrAnalyzer* est donc la plus propice à créer des *bugs* et doit être refactorisée en premier.

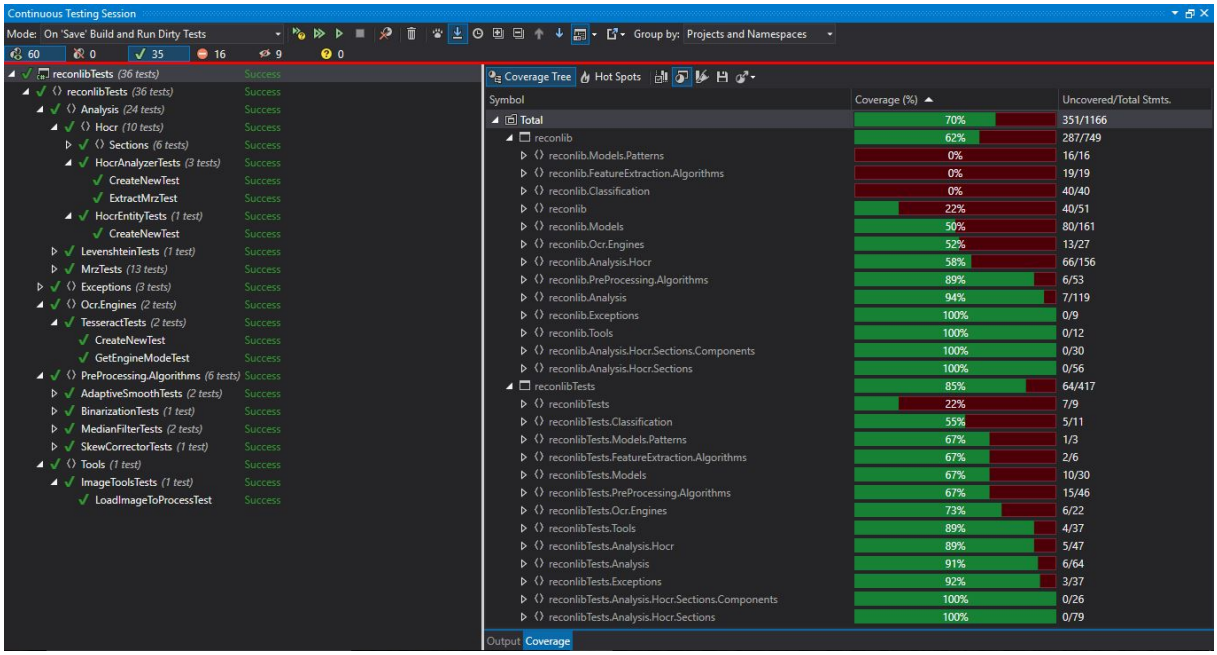


Figure 9 – *Test unitaires : rapport*

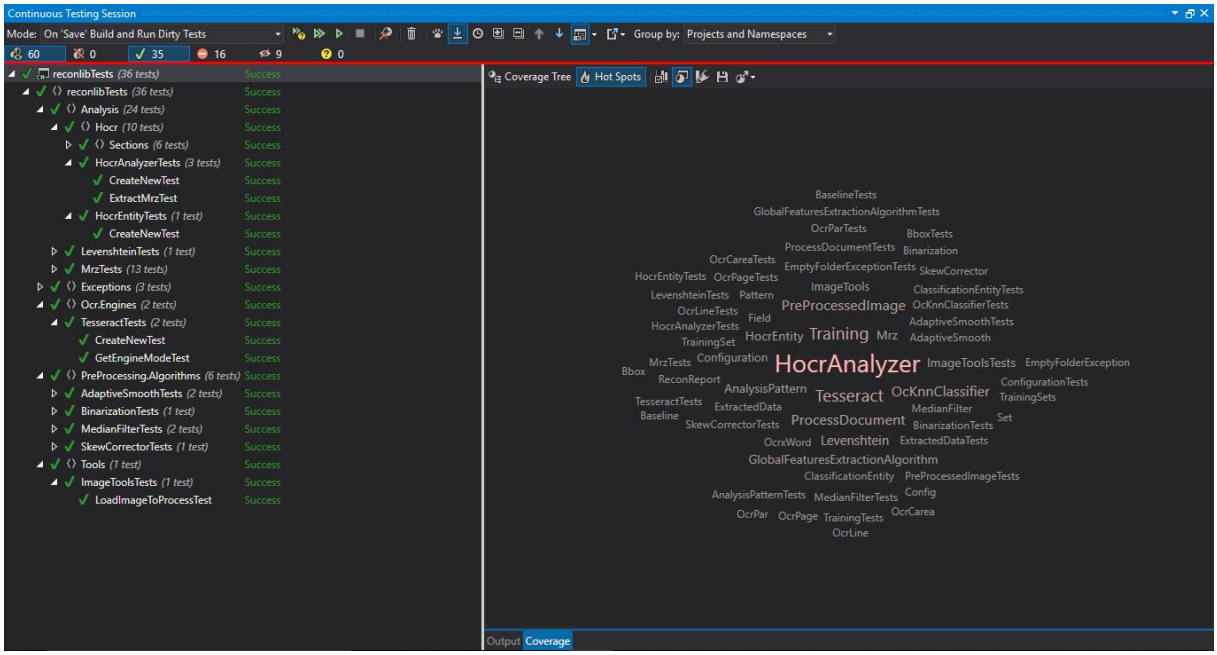


Figure 10 – *Test unitaires : Hotspots*

8 Reproductibilité

Le système développé est fourni sous forme d'une *DLL* [1] ainsi que toutes les dépendances nécessaires. Afin de l'utiliser, il suffit de créer un nouveau projet (console par exemple) puis de référencer la librairie, comme sur la Figure 1.

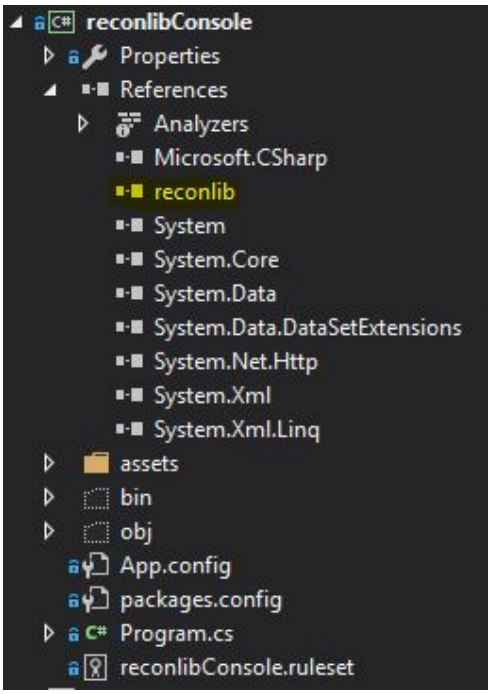


Figure 1 – *Référencer la libraire sous Visual Studio*

Afin de reprendre le projet, il suffit de se référer au chapitre **Mise en Œuvre** et installer les dépendances nécessaires. Il suffit ensuite d'ouvrir/importer le projet (fichier *.sln*) sous Visual Studio et mettre à jour les dépendances, gérées par le gestionnaire de packets *NuGet*.

1. Dynamic Link Library

9 Adaptation du système à des nouveaux documents

La version actuelle du système permet la pris en compte de documents juridiques de type CNI [1]. Il est cependant possible d'adapter le système à la prise en charge d'autres documents. Ci-dessus les étapes nécessaires à la réalisation de cette adaptation.

Création de la base d'apprentissage

Tout d'abord il est necessaire d'avoir une base d'apprentissage du document à intégrer au système. Cette base doit être composée de documents sous format image (tiff de préférence, png ou jpeg également supportés) en niveaux de gris. Cette base d'images doit ensuite être ajoutée au dossier **assets/training_set** de l'application.

Prise en compte de la base par le système

Une fois cette nouvelle base ajouté, il est nécessaire d'indiquer au système le besoin de chargement au démarrage. Pour cela, il faudra modifier le fichier de configuration des bases d'apprentissage, qui se trouve dans **config/training_sets.xml**. Il sera nécessaire d'ajouter une entrée à ce fichier du type **<trainingSet id=" " path=" " pattern=" "/>** :

- **id :** l'identifiant de la base d'apprentissage. Doit être **unique**
- **path :** le nom du dossier où sont stockés les fichiers de la base
- **pattern :** le nom du pattern d'analyse du document

Le pattern d'analyse

Lors du renseignement de la nouvelle base dans le fichier de configuration, un fichier *pattern* est demandé. Ce fichier permet de traduire la structure du document sous format *xml* et renseigne le système sur les éventuelles données à extraire. Les fichiers *pattern* se trouvent dans **config/patterns** et contiennent des entrés du type **<field name=" " label=" " expected=" " delta=" " />** :

- **name :** l'identifiant du champs. Doit être **unique**
- **label :** permet la reconnaissance du label de la donnée à extraire. Il s'agit généralement du dernier mot avant.
- **expected :** le nombre de mots à extraire suite à la reconnaissance du label. Attends des valeurs du type **1** ou **+**

1. Carte Nationale d'Identité

— **delta :** le taux de différence entre le label saisi et celui détecté dans le document. Si valeur à 0, alors le mot doit être exactement identique à celui saisi dans le fichier.

Chaque entrée va ainsi représenter une information à extraire du document.

Redémarrage du service

Une fois les étapes précédentes exécutées, il sera nécessaire de redémarrer le service. Au démarrage le système va détecter la nouvelle base et va itérer parmi les nouvelles images afin d'en extraire les caractéristiques sous format *xml*. Le nouveau document sera désormais pris en compte par le système.

10 Perspectives d'avenir

Du aux contraintes temporelles du projet, le système développé ne représente qu'une base de travail et réflexion pour des évolutions à venir. Les possibilités sont multiples. Pour n'en citer que quelques unes :

— Lier le moteur OCR Tesseract à des bases de données tels que des bases de prénoms ou noms, d'adresses, etc. Cela va permettre d'améliorer grandement les taux de reconnaissance de ce dernier et, par la même occasion, la robustesse globale du système. Des services comme La Poste utilisent ce système depuis plusieurs années afin d'effectuer le tri automatique du courrier.
— Utiliser les statistiques de la langue française afin de corriger les mots mal interprétés
— Rendre le service internationale et permettre la reconnaissance de documents étrangers
— Améliorer le système de traitement des images
— Séparer les fonctionnalités et basculer une partie des traitements sur le système des clients (application mobile) afin de diminuer la charge du serveur.

Troisième partie

Gestion de Projet

11 Méthodologie

La méthode de gestion choisie pour ce projet est le *cycle en V*. La méthode *Agile* aurait pu être utilisée également en définissant un *Sprint* par fonctionnalité. Néanmoins, chaque *Sprint* aurait été de courte durée et nécessiterait la mise en place d'un *Backlog* à chaque début. D'une part, cela aurait pris trop de temps (rédaction des *Backlog*) et des *Sprints* de courte durée n'auraient pas été rentables.

C'est pourquoi la méthode du *cycle en V* à été choisie. De plus, le premier semestre nous à permis de rédiger les spécifications et l'état de l'art, ce qui suit la logique de cette méthodologie. S'en suit une phase de développement, les tests (unitaires puis d'intégration). Enfin la phase recette, qui se fera par la suite. Ces étapes et leur enchainement, comme mentionné précédemment, engendrent de manière naturelle la logique du *cycle en V*.

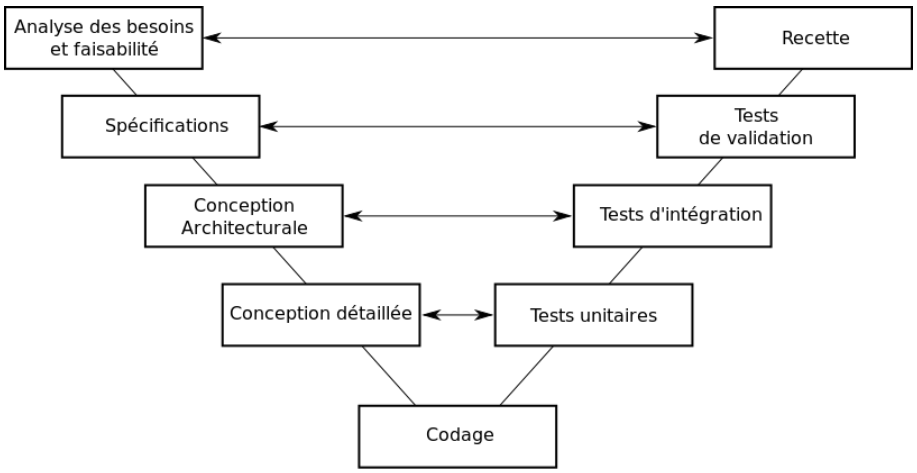


Figure 1 – *Cycle en V : fr.wikipedia.org*

12 Plan de développent

1 Découverte du sujet

Description de la tâche

Le but de cette tâche est d'appréhender le sujet avant le début des recherches bibliographiques, ce qui permettra de mieux cibler ces recherches.

Estimation de charge

Cette tâche nécessite 1 jour.

2 Analyse des besoins

Description de la tâche

L'analyse des besoins est une tâche importante dans tout projet. Elle à pour but d'analyser les besoins exprimés et de mieux diriger les recherches.

Estimation de charge

Cette tâche nécessite 1 jour.

3 Veille technologique et étude bibliographique

Description de la tâche

La veille technologique et l'étude bibliographique est une tâche qui durera tout le long de la phase de recherche. Cette tâche consiste à étudier la littérature et les nouvelles technologies afin de trouver les solutions les plus adaptés au problème de reconnaissance de documents, tout en considérant les contraintes du projet.

Estimation de charge

Cette tâche durera tout le long de la phase de recherche, c'est à dire 22 jours.

4 Rédaction de l'état de l'art

Description de la tâche

L'état de l'art est un document synthétisant les différents travaux existants dans le domaine de la reconnaissance de documents. Rédigé grâce à la veille technologique et à l'étude de la littérature, ce document permettra d'effectuer une analyse des méthodes existantes, ce qui contribuera à la rédaction du cahier des spécifications.

Estimation de charge

Cette tâche durera tout le long de la phase de recherche, c'est à dire 22 jours.

Livrables

Cette tâche fera l'objet d'un livrable, à rendre pour le 09/12/2016.

5 Rédaction du cahier de spécification

Description de la tâche

Le cahier de spécifications est un document regroupant l'ensemble des spécifications techniques et fonctionnelles relatives au projet à développer. Ce document contient ainsi des informations tels l'environnement du projet, ses objectifs ou le descriptif exhaustif des fonctionnalités à développer tout au long du projet, et devra être validé par la MOE et par la MOA avant le début de la phase de développement.

Estimation de charge

Cette tâche nécessitera 18 jours.

Contraintes temporelles

La cahier de spécifications doit être validé avant la phase de développement, c'est à dire avant le 04/01/2017

Livrables

Cette tâche fera l'objet d'un livrable, à rendre pour validation avant le 24/11/2016, et validé avant le 04/01/2017.

6 Validation du portage *Mono*

Description de la tâche

Cette tâche consiste à faire valider le portage de *Mono* sous la distribution Linux Debian 8 (jessie). L'objectif est ainsi de confirmer la faisabilité de son installation, sa stabilité, et le bon fonctionnement du *framework* .NET. Pour cela une VM comportant une installation de Linux Debian 8 (jessie) devra être mise en place.

Estimation de charge

Cette tâche nécessitera 1 jour.

Contraintes temporelles

Cette tâche devra être accomplie avant la fin de la rédaction du cahier de spécifications.

Livrables

Le livrable attendu pour cette tâche est une étude de faisabilité sous forme d'un rapport, validant le bon fonctionnement de *Mono* dans l'environnement d'exécution du projet.

7 Analyse et choix des algorithmes à utiliser

Description de la tâche

Cette tâche consiste à analyser l'état de l'art et choisir les meilleurs algorithmes à utiliser dans le système.

Estimation de charge

Cette tâche nécessitera 8 jour.

Contraintes temporelles

Le choix des algorithmes devra être effectué avant le début du développement de la fonctionnalité de pré-traitement.

8 Prise en main de *Docker* et *RabbitMQ*

Description de la tâche

Docker et *RabbitMQ* sont deux technologies utilisées par le service FoxNot, et il est impératif de les utiliser également lors de la réalisation de ce projet, afin de maintenir une compatibilité maximale entre les deux.

Docker est un logiciel qui permet le déploiement d'applications dans des conteneurs virtuels. Ces conteneurs possèdent toutes les dépendances nécessaires au bon fonctionnement du logiciel à déployer.

RabbitMQ est un logiciel qui permet la gestion de files aux quelles des applications peuvent se connecter afin de récupérer des *messages*. Ces *messages* peuvent contenir de nombreuses informations tels des indications sur un processus qu'une application devra exécuter ou simplement des données à transférer entre applications.

Estimation de charge

Cette tâche nécessitera 2 jours.

Contraintes temporelles

Cette tâche devra être effectuée avant le début du développement des *workers*.

Livrables

Le livrable attendu pour cette tâche est une étude de faisabilité sous forme d'un rapport, validant le bon fonctionnement des deux technologies dans l'environnement d'exécution du projet.

9 Préparation de la base d'apprentissage

Description de la tâche

La base d'apprentissage est un ensemble d'images utilisées pour enseigner un algorithme de classification (ex. SVM). Cette base est ainsi primordiale lors de la phase de reconnaissance de documents. Le but de cette tâche est de regrouper un ensemble de documents dématérialisés (scan ou photo) des pièces d'identité à reconnaître durant le projet, afin de constituer une base d'images d'apprentissage.

Estimation de charge

Cette tâche nécessitera 1 jour.

Contraintes temporelles

La base d'apprentissage devra être constituée avant la fin du développement du *worker* de reconnaissance de documents.

Livrables

Le livrable attendu pour cette tâche est la base d'apprentissage mise en place.

10 Développement de la fonctionnalité de pré-traitement des images

Description de la tâche

Le but de cette tâche est de mettre en place la liaison avec les librairies de traitement et développer les algorithmes de pré-traitement qui ne sont pas disponibles dans ces librairies. La fonctionnalité finale devra être capable de prendre en compte une image, appliquer l'ensemble des traitements définis et de fournir l'image traitée en sortie.

Estimation de charge

Cette tâche nécessitera 9 jours.

Contraintes temporelles

Cette tâche devra être effectuée avant la tâche de chargement des images.

Livrables

Cette tâche fera l'objet d'un livrable. Ce livrable sera constitué du *worker* développé et d'un compte-rendu des tests effectués.

11 Mise en place du bouchon de pré-traitement

Description de la tâche

Mettre en place un système capable de générer des images dont les caractéristiques correspondent aux images pré-traitées. Cela aura pour but de substituer temporairement à la fonctionnalité de pré-traitement, en cas de problème rencontré lors de son développement, afin de ne pas retarder le développement des autres fonctionnalités.

Estimation de charge

Cette tâche nécessitera 1 jours.

Contraintes temporelles

Cette tâche devra être effectuée avant la tâche de chargement des images.

12 Développement du *worker* de chargement et pré-traitement de l'image

Description de la tâche

La tâche consistera à développer le *worker* de chargement de l'image et mettre en place la liaison avec la fonctionnalité de pré-traitement des images.

Estimation de charge

Cette tâche nécessitera 5 jours.

Contraintes temporelles

Cette tâche devra être effectuée avant les autres tâches nécessitant une image en entrée.

Livrables

Cette tâche fera l'objet d'un livrable. Ce livrable sera constitué du *worker* développé et d'un compte-rendu des tests effectués.

13 Développement du *worker* d'extraction de caractéristiques

Description de la tâche

Cette tâche consiste à développer le *worker* d'extraction de caractéristiques, c'est à dire, la liaison avec les bibliothèques externes et la mise en place des algorithmes nécessaires qui ne sont pas disponibles dans ces bibliothèques.

Estimation de charge

Cette tâche nécessitera 10 jours.

Contraintes temporelles

Cette tâche doit être effectuée avant le développement du *worker* de classification de documents.

Livrables

Cette tâche fera l'objet d'un livrable. Ce livrable sera constitué du *worker* développé et d'un compte-rendu des tests effectués.

14 Mise en place du bouchon de génération de caractéristiques

Description de la tâche

Mettre en place un système capable de générer des vecteurs de caractéristiques, correspondant à ceux générés par le *worker* d'extraction de caractéristiques. Cela aura pour but de substituer temporairement le *worker* d'extraction, en cas de problème rencontré lors de son développement, afin de ne pas retarder le développement des autres fonctionnalités.

Estimation de charge

Cette tâche nécessitera 1 jours.

Contraintes temporelles

Cette tâche doit être effectuée avant le développement du *worker* de classification de documents.

15 Développement du *worker* de classification de documents

Description de la tâche

Cette tâche consiste à développer les algorithmes de classification et mettre en place les liaisons avec les bibliothèques externes.

Estimation de charge

Cette tâche nécessitera 9 jours.

Contraintes temporelles

Cette tâche devra être effectuée avant le développement du *worker* d'analyse sémantique.

Livrables

Cette tâche fera l'objet d'un livrable. Ce livrable sera constitué du *worker* développé et d'un compte-rendu des tests effectués.

16 Mise en place du bouchon de génération de classifications

Description de la tâche

Mettre en place un système capable de générer des classifications de documents, correspondant à ceux générés par le *worker* dde classification de documents. Cela aura pour but de substituer temporairement le *worker* de classification, en cas de problème rencontré lors de son développement, afin de ne pas retarder le développement des autres fonctionnalités.

Estimation de charge

Cette tâche nécessitera 1 jours.

Contraintes temporelles

Cette tâche devra être effectuée avant le développement du *worker* d'analyse sémantique.

17 Développement du *worker* de lecture par OCR

Description de la tâche

Le but de cette tâche est de créer un environnement d'exécution pour le système de OCR externe (tesseract), afin que ce dernier soit utilisable par le module de reconnaissance des documents.

Estimation de charge

Cette tâche nécessitera 5 jours.

Contraintes temporelles

Cette tâche devra être effectuée avant le développement du *worker* d'analyse sémantique.

Livrables

Cette tâche fera l'objet d'un livrable. Ce livrable sera constitué du *worker* développé et d'un compte-rendu des tests effectués.

18 Mise en place du bouchon de génération de texte

Description de la tâche

Mettre en place un système capable de générer des fichiers texte contenant des données identiques à celles d'un document juridique. Cela aura pour but de substituer temporairement le *worker* de lecture par OCR, en cas de problème rencontré lors de son développement, afin de ne pas retarder le développement des autres fonctionnalités.

Estimation de charge

Cette tâche nécessitera 1 jours.

Contraintes temporelles

Cette tâche devra être effectuée avant le développement du *worker* d'analyse sémantique.

19 Développement du *worker* d'analyse et extraction sémantique

Description de la tâche

Cette tâche consiste à créer un système permettant d'analyser sémantiquement le fichier texte fourni par le *worker* de lecture par OCR et, dépendant de la classe du document traité (fourni par le *worker* de classification), être capable d'extraire les données d'intérêt (nom, prénom, adresses...) contenues dans ce document.

Estimation de charge

Cette tâche nécessitera 11 jours.

Contraintes temporelles

Cette tâche devra être effectuée avant la validation du POC.

Livrables

Cette tâche fera l'objet d'un livrable. Ce livrable sera constitué du *worker* développé et d'un compte-rendu des tests effectués.

20 Développement du *worker* d'apprentissage de nouveaux cas d'utilisation

Description de la tâche

Cette tâche consiste à développer un système permettant de renseigner la base d'apprentissage avec de nouveaux cas d'utilisation (nouvelles classes de documents), afin de rendre le module évolutif.

Estimation de charge

Cette tâche nécessitera 11 jours.

Contraintes temporelles

Cette tâche devra être effectuée avant la fin du projet.

Livrables

Cette tâche fera l'objet d'un livrable. Ce livrable sera constitué du *worker* développé et d'un compte-rendu des tests effectués.

21 Validation du POC

Description de la tâche

Cette tâche consiste à faire valider les modules développés pendant le projet. Cette validation se fera grâce à un ensemble de documents plus complexes que ceux utilisés pendant la phase de développement (DPE par exemple).

Estimation de charge

La mise en place de la validation du POC nécessite 7 jours.

Contraintes temporelles

Cette tâche devra être effectuée avant la fin du projet.

Livrables

La tâche fera l'objet d'un livrable correspondant à un rapport de validation de la preuve de concept.

22 Rédaction du rapport

Description de la tâche

Cette tâche consiste à continuer la rédaction du rapport débuté avec la rédaction de l'état de l'art et du cahier de spécifications. Ce document contiendra toutes les informations relatives au déroulement du projet.

Estimation de charge

Cette tâche se déroulera tout au long du projet.

Contraintes temporelles

Cette tâche devra être réalisée avant la fin du projet.

Livrables

La tâche fera l'objet d'un livrable correspondant au rapport de projet.

23 Préparation de la soutenance de fin de projet

Description de la tâche

Cette tâche consiste à préparer les supports nécessaires pour la soutenance de fin de projet.

Estimation de charge

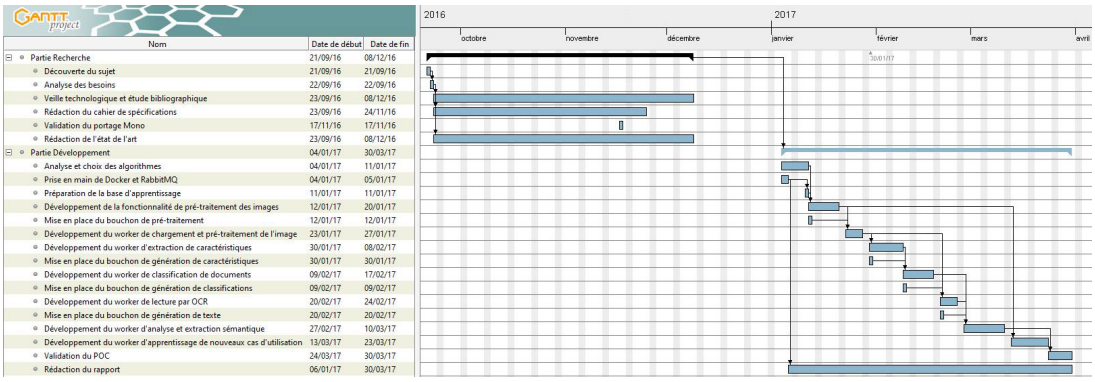
Cette tâche nécessitera 1 jour.

Livrables

La tâche fera l'objet d'un livrable correspondant aux supports nécessaires pour la soutenance de fin de projet.

13 Plannings

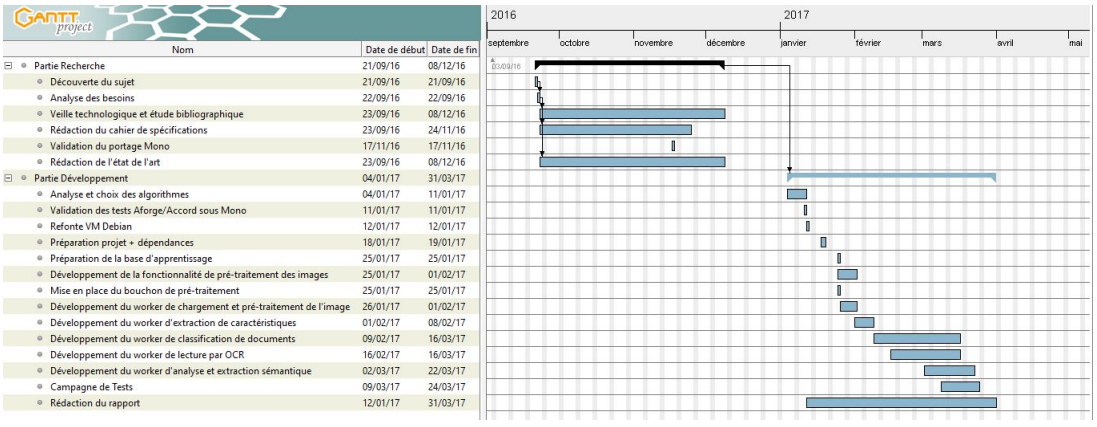
1 Planning prévisionnel



Nom	Date de début	Date de fin
Partie Recherche	21/09/16	08/12/16
Découverte du sujet	21/09/16	21/09/16
Analyse des besoins	22/09/16	22/09/16
Veille technologique et étude bibliographique	23/09/16	08/12/16
Rédaction du cahier de spécifications	23/09/16	24/11/16
Validation du portage Mono	17/11/16	17/11/16
Rédaction de l'état de l'art	23/09/16	08/12/16
Partie Développement	04/01/17	30/03/17
Analyse et choix des algorithmes	04/01/17	11/01/17
Prise en main de Docker et RabbitMQ	04/01/17	05/01/17
Préparation de la base d'apprentissage	11/01/17	11/01/17
Développement de la fonctionnalité de pré-traitement des images	12/01/17	20/01/17
Mise en place du bouchon de pré-traitement	12/01/17	12/01/17
Développement du worker de chargement et pré-traitement de l'image	23/01/17	27/01/17
Développement du worker d'extraction de caractéristiques	30/01/17	08/02/17
Mise en place du bouchon de génération de caractéristiques	30/01/17	30/01/17
Développement du worker de classification de documents	09/02/17	17/02/17
Mise en place du bouchon de génération de classifications	09/02/17	09/02/17
Développement du worker de lecture par OCR	20/02/17	24/02/17
Mise en place du bouchon de génération de texte	20/02/17	20/02/17
Développement du worker d'analyse et extraction sémantique	27/02/17	10/03/17
Développement du worker d'apprentissage de nouveaux cas d'utilisation	13/03/17	23/03/17
Validation du POC	24/03/17	30/03/17
Rédaction du rapport	06/01/17	30/03/17

Figure 1 – *Planning prévisionnel du projet -* ***en annexes***

2 Planning réel



Nom	Date de début	Date de fin
Partie Recherche	21/09/16	08/12/16
Découverte du sujet	21/09/16	21/09/16
Analyse des besoins	22/09/16	22/09/16
Veille technologique et étude bibliographique	23/09/16	08/12/16
Rédaction du cahier de spécifications	23/09/16	24/11/16
Validation du portage Mono	17/11/16	17/11/16
Rédaction de l'état de l'art	23/09/16	08/12/16
Partie Développement	04/01/17	31/03/17
Analyse et choix des algorithmes	04/01/17	11/01/17
Validation des tests Aforge/Accord sous Mono	11/01/17	11/01/17
Refonte VM Debian	12/01/17	12/01/17
Préparation projet + dépendances	18/01/17	19/01/17
Préparation de la base d'apprentissage	25/01/17	25/01/17
Développement de la fonctionnalité de pré-traitement des images	25/01/17	01/02/17
Mise en place du bouchon de pré-traitement	25/01/17	25/01/17
Développement du worker de chargement et pré-traitement de l'image	26/01/17	01/02/17
Développement du worker d'extraction de caractéristiques	01/02/17	08/02/17
Développement du worker de classification de documents	09/02/17	16/03/17
Développement du worker de lecture par OCR	16/02/17	16/03/17
Développement du worker d'analyse et extraction sémantique	02/03/17	22/03/17
Campagne de Tests	09/03/17	24/03/17
Rédaction du rapport	12/01/17	31/03/17

Figure 2 – *Planning réel du projet -* ***en annexes***

14 Analyse

1 Analyse des plannings

Comme nous pouvons le remarquer sur les Figure 1 (Chapitre 13) et Figure 2 (Chapitre 13), le planning réel du semestre 10 diffère légèrement du planning prévisionnel établi. Cela est du au fait que certaines fonctionnalités ont mis moins de temps à être développées que prévu alors que d'autres fonctionnalités ont mis beaucoup plus de temps.

Nous n'allons pas nous attarder sur les fonctionnalités qui ont mis moins de temps que prévu car cela est une bonne chose en soi et la seule remarque possible serait une sur-estimation du temps de développement de ces tâches.

Les plus constructives à commenter restent les tâches qui ont mis *beaucoup* plus de temps que prévu à être effectuées. Les raisons de ce retard seront expliquées dans la suite du chapitre.

2 Analyse de faisabilité

Lors de la rédaction du cahier de spécifications, la définition du périmètre du projet à pris en compte les contraintes de temps. Malgré que cette prise en compte ne soit que théorique, cela à néanmoins permis de commencer le projet avec une base d'objectifs réalisables dans le temps imparti.

Concernant les ressources, cela à été plus délicat. En effet, une partie des ressources nécessaires au projet n'a pas pu être livrée à temps par le client. Comme expliqué dans **Suivi de projet**, l'obtention des documents juridiques nécessaires au bon déroulement du projet à pris plus de temps que prévu et à causé un retard sur le projet. L'équipe RFAI à néanmoins pu fournir un jeu de documents ce qui à permis de combler cet aléas.

Cependant, les documents fournis par l'équipe ne sont pas représentatifs des documents que le client peut utiliser. Cela à donc engendré une modification des objectifs, notamment concernant le format des documents à prendre en compte. De plus, ces documents ne sont que d'un seul type (CNI). En absence de documents du type "passeport français", l'objectif initial de prise en charge des CNI **et** passeports français à du être modifié.

Finalement, du à un aléas imprévisible (expliqué également dans **Suivi de projet**), un retard considérable est survenu en fin de projet. Cela joint à un mauvais format des documents

juridiques de type *DPE* [1] fournis par le client à engendré la suppression de l'objectif *Validation du POC*.

3 Analyse de risque

La prise de risque à en partie été gérée par la mise en place de bouchons. Ces bouchons avaient pour objectif de générer des données de remplacement permettant de continuer le développement d'autres fonctionnalités dépendantes si un échec survenait sur une fonctionnalité en particulier. De plus, seulement quelques fonctionnalités étaient réellement critiques, comme la fonctionnalité de pré-traitement (nécessaire au bon fonctionnement de toutes les autres fonctionnalités) ou bien celle de traitement par OCR. Cela rendait le développement parallèle possible ce qui permettait de réduire les risques d'échec en cas d'aléas. Ainsi, si une tâche dérivait dans le temps, il était possible de continuer le développement d'autres fonctionnalités en attendant de trouver une solution viable. Cela s'est d'ailleurs avéré décisif lors des dernières étapes du projet comme explique dans **Suivi de projet**.

Cependant, au cours du projet, il s'est avéré que la mise en place de certains bouchons n'était pas possible car trop dépendants de certaines librairies. Par manque d'exemples viables, il n'était pas possible de créer un bouchon *fonctionnel* tant que l'intégration de la bibliothèque correspondante n'était pas effectuée. Cela augmentait de manière considérable le risque d'échec mais ne pouvait pas être traité autrement.

1. Diagnostique de performance énergétique

15 Suivi de projet

L'avancée du projet à été suivie par biais de compte-rendus hebdomadaires et de réunions périodiques avec les encadrants. Cela à parfois permis de guider l'avancée du projet et d'apporter d'autres pistes de réflexion. De plus, ces rencontres ont également permis de valider le projet au fur et à mesure de son développement, tant du coté des encadrants que du client.

Pendant la phase de développent, deux aléas sont survenus. Tout d'abord concernant les ressources nécessaires au développement du projet : les documents juridiques. Ces documents présentaient le coeur même du projet et étaient essentiels pour son bon déroulement. Cependant, du à la nature sensible des données contenues dans ces documents (CNI et passeports), leur obtention de part l'entreprise cliente à pris plus de temps que prévu. Afin de combler ce problème, l'équipe RFAI à pu fournir des exemples de tests ce qui à permis de mener à bien les objectifs posés au départ.

Le second aléas survenu concernait la bibliothèque de classification fournie par l'équipe RFAI. En effet, certainement du à des versions incompatibles, une grande phase de déboggage à du avoir lieu afin de trouver la source des erreurs engendrés par l'intégration de cette librairie. Avec l'aide de M. Jean-Yves Ramel, une solution à pu être trouvée et l'intégration correctement effectué. Cela à néanmoins causé un retard important du projet.

Du à leur nature, ces deux aléas n'ont pas pu être prévus à l'avance.

16 Outils de gestion

Afin de mener à bien la gestion de ce projet, plusieurs outils ont été utilisés :

- **GanttProject :** Cet outils à permis de rédiger et de maintenir les plannings du projet. C'est ainsi un outils indispensable pour la gestion.
- **GitLab :** Servant également de *repository*, GitLab propose des services de gestion (liste des tâches, métriques). Cela à été utile afin de suivre la progression du projet.
- **GitKraken :** Ce logiciel permet de rendre la mécanique *Git* plus facilement utilisable par moyen d'une interface graphique ergonomique et des raccourcis d'utilisation.
- **Trello :** Cette plateforme permet d'organiser les tâches à effectuer pendant le projet par moyen de tableaux et de listes à cocher. Cela à permis de garder un fil conducteur tout au long du projet et de ne pas oublier les tâches les plus importantes à effectuer.
- **Sonar :** Cet outil de mesure de qualité de code à permis une détection en temps les éventuels problèmes liés à la rédaction de code et donc de prévoir à l'avance des corrections à apporter.

Quatrième partie

Bilan

17 Point sur l'avancement

1 Partie recherche

Concernant la partie recherche, le planning à été suivi et aucun retard n'est à constater. La seule partie qui ne pas été effectuée totalement est l'analyse, mais cela est volontaire comme expliqué en préambule de ce chapitre.

2 Partie développement

Les objectifs posés pour la partie développement ont, dans leur majorité, été atteints. Comme expliqué précédemment dans ce rapport, des aléas non prévisibles ont été rencontrés ce qui à retardé l'avancement du projet. Les objectifs principaux ont néanmoins été réalisés et une version fonctionnelle à pu être livrée au client.

18 Résultat final

Clause de non-responsabilité : *Du à la nature sensible des informations contenues dans les documents juridiques de type CNI, et par des raisons de confidentialité, le document présenté dans ce chapitre contient des données faussées et ne représente en aucun cas, des informations relatives à une personne physique. La modification de ce document n'a été effectuée que pour des raisons de démonstration.*

Figure 1 – *Exemple de résultat final délivré par la librairie*

L'objectif de la bibliothèque développée tout au long de ce projet est de, a partir d'un document juridique donné, être capable de reconnaître le type du document et d'en extraire des informations utiles.

Comme mentionné précédemment, du à certains aléas, seulement les documents de type CNI sont pour le moment pris en charge par la bibliothèque. Néanmoins, un effort à été effectué afin de rendre cette bibliothèque évolutive (voir **Adaptation du système à des nouveaux documents**.

La Figure 1 démontre un exemple de résultat obtenu lorsque la CNI à gauche de l'image est fournie au service.

Tout d'abord nous remarquons que le document est bien identifié comme étant de type CNI (*Classified as cni*). Ensuite nous remarquons également que les données extraites directement du document possèdent pour la plupart des erreurs. Cela est du, d'une part, à la qualité de l'image (les liaisons entre les composantes des lettres sont faibles ce qui rends leur détection par le moteur OCR difficile), et d'une autre part à la configuration de Tesseract. Nous remarquons néanmoins que le prénom à pu être extrait et est conforme.

Concernant le MRZ, l'ensemble des informations à pu être extrait et les données sont conformes sauf la date de naissance et une erreur dans le prénom. Ces erreurs sont en parti du au montage des informations dans la pièce (rappelons qu'il s'agit d'une pièce fictive). Finalement, il nous est également renseigné que la structure du MRZ est bonne mais que ses données ne sont pas valides (à nouveau, il s'agit d'une pièce fictive).

D'autres validations sont également effectuées par calcul des clés (ID et date de naissance) et des indices de confiance d'extraction sont fournis par le moteur OCR.

Afin de fournir ces informations, le système va commencer par charger l'image et confirmer sa compatibilité. Ensuite, des filtres et algorithmes de pré-traitement sont appliqués à l'image afin de la rendre exploitable (binarization) et afin d'améliorer les taux de reconnaissance du moteur OCR (*Skew Corrector*, filtre médian). Une fois cela effectué, l'image traitée est récupérée par la fonctionnalité de classification qui fournira la classe du document présent dans l'image. Parallèlement, le moteur OCR va traiter l'image et extraire l'ensemble des informations contenues sous format HOCR. La classification et les données HOCR sont ensuite traités par la fonctionnalité d'analyse et extraction sémantique qui va extraire les données d'intérêt basé sur la classification du document.

Le résultat est fourni sous forme d'un rapport utilisable ensuite par l'utilisateur. Ici, un programme *Console* initialise le service et traite une image donnée. Le rapport est ensuite récupéré et le résultat affiché sur la console.

19 Conclusion

En conclusion, l'analyse et l'étude des solutions actuelles nécessaires à la rédaction de l'état de l'art m'a permis de me confronter à une méthodologie de recherche que je n'avais jamais utilisé auparavant. Cela à donc été très bénéfique car grâce à cela j'ai pu acquérir des nouvelles compétences en recherche et réflexion.

La partie développement à été l'occasion de mettre en avant les nouvelles connaissances acquises lors de la phase de recherche. Cette partie à également été l'occasion pour moi de reprendre en main une technologie que je n'avais pas utilisé depuis longtemps (C# et .Net) et de me former à d'autres technologies, comme Tesseract par exemple.

Même si cette phase ne s'est pas déroulée comme planifié, les objectifs principaux ont néanmoins pu être atteints et une version fonctionnelle à pu être livrée au client.

Dans l'ensemble, le projet s'est montré très enrichissant car ça m'a permis de me confronter à des problématiques jusque la jamais rencontrés. Cela à été très formateur. Concernant le sujet du projet, il m'a permis de traiter un domaine nouveau et à été l'occasion de monter en compétences dans un domaine en plein essor aujourd'hui.

Cinquième partie

Glossaire

- **CNI :** Carte Nationale d'Identité
- **DPE :** Le Diagnostic de Performance Énergétique est un document faisant partie du dossier de diagnostics techniques et est réalisé en France sur l'ensemble des biens immobiliers. Ce document à pour but d'informer un propriétaire ou locataire des niveaux de consommation énergétique de son habitation.
- **FIFO :** *First In First Out* ou premier arrivé, premier servi.
- **hOCR :** Format de fichier basé sur la structure HTML et servant à l'enregistrement des données fournies par un logiciel d'OCR. Les spécifications peuvent être consultées ici [WWW32]
- **MOA :** La Maîtrise d'ouvrage est l'entité (personne ou entreprise) qui exprime le besoin et les objectifs du projet à développer.
- **MOE :** La Maîtrise d'œuvre est l'entité qui intervient lors de la réalisation du projet. C'est cette entité qui décrit les spécifications fonctionnelles et non-fonctionnelles du projet et qui réalise son développement.
- **MRZ :** *Machine Readable Zone*
- **OCR :** *Optical Character Recognition* ou Reconnaissance Optique de Caractères
- **POC :** *Proof of Concept* ou Preuve de Concept ou Démonstration de Faisabilité, est une pratique servant à démonter la faisabilité d'une méthode ou technique par moyen de réalisations expérimentales.
- **Structure de type *Manhattan* :** Structure de document composée de zones carrées séparées par des espaces blancs perpendiculaires.
- **Structure de type *Non-Manhattan* :** Inverse de structure de type *Manhattan*. Le document ne se compose pas seulement de zones carrées et séparations perpendiculaires, et peut contenir des zones de forme organique ou arrondies.
- **Système de versionning :** Système permettant de gérer les versions d'un logiciel. Ce système permet ainsi de sauvegarder toutes les modifications apportées à un ensemble de fichiers de manière chronologique.
- **VM :** *Virtual Machine* ou Machine Virtuelle
- **Worker :** Module de l'application qui effectue un traitement qui lui est propre.

Sixième partie

Annexes

1 Diagramme de l'architecture générale du système

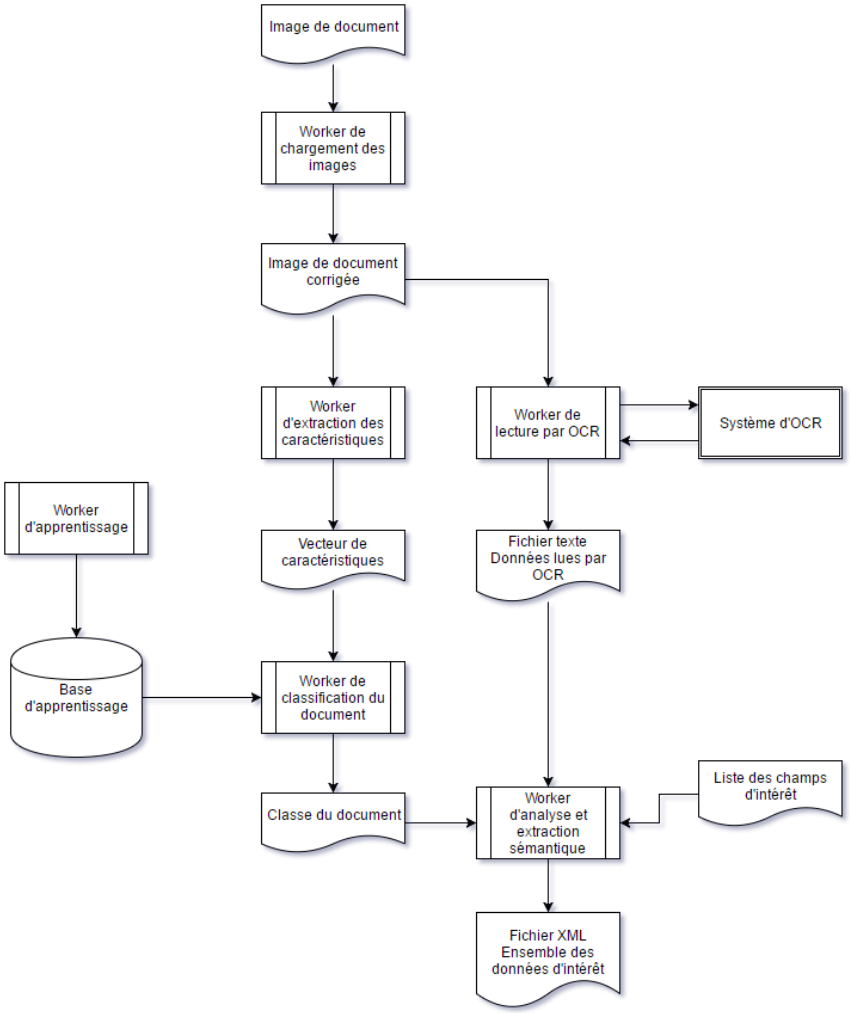


Figure 1 – *Architecture générale du système*

2 Spécifications

2.1 Spécifications fonctionnelles

Les fonctionnalités présentées dans ce chapitre correspondent aux *workers* à développer pendant ce projet. Il faudra noter que les algorithmes présentés dans ce chapitre son simplifiés. En effet, ces algorithmes seront développés suite à des tests qui permettront de définir les outils les plus adaptés aux tâches concernées. Le but ici c'est de définir la structure générale et de donner un aperçu du fonctionnement des *workers*.

2.1.1 *Worker* de chargement et pré-traitement de l'image

Présentation de la fonctionnalité

- **Rôle :** Chargement d'une image (scan ou photo) contenant un document et application des pré-traitements nécessaires (voir fonctionnalité de pré-traitement).
- **Priorité :** Haute

Description de la fonctionnalité

— **Entrées/Sorties :** En entrée cette fonctionnalité prendra une image (scan ou photo) contenant un document à traiter. En sortie elle fournira une image, correspondant à l'image d'entrée à laquelle seront appliqués les traitement de la fonctionnalité de pré-traitement des images.

— **Interactions :** Cette fonctionnalité va interagir avec la fonctionnalité de pré-traitement des images. Cette fonctionnalité va envoyer l'image chargée à la fonctionnalité de pré-traitement des images (passage par référence) et va récupérer l'image traitée (s'agissant d'un passage par référence, c'est l'instance de l'image chargée qui sera directement modifiée).

— **Traitement associé :**

```
Data : Document Image
image ← load(document_image);
if image ≠ null then
    image ← pre_processing(image);
    save(image);
else
    throw Exception;
end
```

Algorithme 1 : Chargement et pré-traitement de l'image

— **Gestion des erreurs :** La gestion des erreurs sera faite par renvoi d'exceptions.

2.1.2 Fonctionnalité de pré-traitement des images

Présentation de la fonctionnalité

- **Rôle :** Application d'un ensemble de traitements à une image donnée afin que celle-ci soit utilisable par l'ensemble des *workers* (binarization, recadrage, rotation, netteté).
- **Priorité :** Haute

Description de la fonctionnalité

— **Entrées/Sorties :** En entrée cette fonctionnalité prendra une image (scan ou photo) contenant un document à traiter. En sortie elle fournira une image, correspondant à l'image d'entrée à laquelle seront appliqués des algorithmes de pré-traitement tels la binarization, des transformations (rotation, recadrage) ou l'amélioration de la netteté.
— **Interactions :** Cette fonctionnalité va interagir avec des bibliothèques externes (accord.net, aforge.net) apportant les algorithmes nécessaires au traitement de l'image.
— **Traitement associé :** Les algorithmes à appliquer seront à définir suite à une phase de tests sur un ensemble de documents, qui permettra de définir les traitements les plus adaptés.

2.1.3 *Worker* d'extraction de caractéristiques

Présentation de la fonctionnalité

- **Rôle :** Utilisation d'algorithmes d'extraction de caractéristiques afin d'extraire les caractéristiques de l'image chargée sous forme d'un vecteur.
- **Priorité :** Haute

Description de la fonctionnalité

— **Entrées/Sorties :** En entrée cette fonctionnalité prendra une image (scan ou photo) contenant un document à traiter. En sortie le fonctionnalité fournira un vecteur contenant l'ensemble des caractéristiques extraites. Les caractéristiques dépendront de l'algorithme/-méthode utilisée, à définir dans la suite du projet.
— **Interactions :** Cette fonctionnalité va interagir avec des bibliothèques externes (aforge.net, accord.net) apportant l'implémentation des algorithmes d'extraction. Les algorithmes à utiliser seront définis par la suite.
— **Traitement associé :**

```
Data : Pre-processed document image
image ← load(document_image);
if image ≠ null then
    vector ← apply_extraction_algorithm(image);
    save(vector);
else
    throw Exception;
end
```

Algorithme 2 : Extraction des caractéristiques

— **Gestion des erreurs :** La gestion des erreurs sera faite par renvoi d'exceptions.

2.1.4 *Worker* de classification de documents

Présentation de la fonctionnalité

- **Rôle :** Prise en charge du vecteur de caractéristiques généré par le *worker* d'extraction de caractéristiques et mise en relation avec la base d'apprentissage afin d'établir une classification du document traité.
- **Priorité :** Haute

Description de la fonctionnalité

— **Entrées/Sorties :** En entrée cette fonctionnalité prendra un vecteur de caractéristiques (fourni par la fonctionnalité d'extraction de caractéristiques). En sortie le fonctionnalité fournira une classification du document utilisé pour la génération du vecteur de caractéristiques (type de document : carte d'identité, passeport, DPE...).
— **Interactions :** Cette fonctionnalité va interagir avec des bibliothèques externes (aforge.net, accord.net) apportant l'implémentation des algorithmes de classification. De plus, la fonctionnalité va interagir avec la base d'apprentissage, nécessaire lors de la phase de classification.
— **Traitement associé :**

```
Data : Caracteristics vector
if vector ≠ null then
    db ← get_learning_DB();
    classification ← apply_classification_algorithm(vector, db);
    save(classification);
else
    throw Exception;
end
```

Algorithme 3 : Classification

— **Gestion des erreurs :** La gestion des erreurs sera faite par renvoi d'exceptions.

2.1.5 *Worker* de lecture par OCR

Présentation de la fonctionnalité

- **Rôle :** Utilisation d'un système d'OCR (tesseract) afin d'effectuer une lecture du document et sauvegarde des données lues sous forme d'un fichier texte.
- **Priorité :** Haute

Description de la fonctionnalité

— **Entrées/Sorties :** En entrée cette fonctionnalité prendra une image contenant un document et pré-traitée par la fonctionnalité de pré-traitement. En sortie cette fonctionnalité fournira un fichier texte contenant l'ensemble des données lue par le système d'OCR sur le document fourni.
— **Interactions :** Cette fonctionnalité va interagir avec un système d'OCR externe (tesseract).
— **Traitement associé :**

```
Data : Pre-processed document image
if image ≠ null then
    data ← apply_OCR(image);
    data_file ← format(data);
    save(data_file);
else
    throw Exception;
end
```

Algorithme 4 : Lecture OCR

— **Gestion des erreurs :** La gestion des erreurs sera faite par renvoi d'exceptions.

2.1.6 *Worker* d'analyse et extraction sémantique

Présentation de la fonctionnalité

- **Rôle :** Analyse sémantique des données lues par OCR et extraction des données d'intérêt (définies selon la classe du document) sous forme d'un fichier XML.
- **Priorité :** Haute

Description de la fonctionnalité

— **Entrées/Sorties :** En entrée cette fonctionnalité prendra un fichier texte (fourni par la fonctionnalité de lecture par OCR), la classe du document (fournie par la fonctionnalité de classification). La classe du document permettra de récupérer une liste indicative contenant les informations à rechercher dans le document (dépendante de la classe du document). En sortie cette fonctionnalité fournira un fichier XML contenant l'ensemble des données d'intérêt extraites par la fonctionnalité, par rapport à la liste des informations à rechercher (nom, prénom, dates, adresses...).
— **Interactions :** Cette fonctionnalité n'interagit avec aucun autre composant.
— **Traitement associé :**

```
Data : OCR text file
Data : Document class
if ocr_text_file ≠ empty ∧ document_class ≠ null then
    researched_data_list ← get_list(document_class);
    interest_data ← key_value_list;
    for term ∈ researched_data_list do
        if ocr_text_file.indexOf(term) ≥ 0 then
            interest_data[term] ← data;
        end
    end
    xml_file ← format(interest_data);
    save(xml_file);
else
    throw Exception;
end
```

Algorithme 5 : Analyse et extraction sémantique

— **Gestion des erreurs :** La gestion des erreurs sera faite par renvoi d'exceptions.

2.1.7 *Worker* d'apprentissage de nouveaux cas d'utilisation

Présentation de la fonctionnalité

- **Rôle :** Permettre la population de la base d'apprentissage avec de nouveaux cas d'utilisation (de nouvelles classes) sous forme d'un ensemble d'images exploitables par le système.
- **Priorité :** Moyenne

Description de la fonctionnalité

— **Entrées/Sorties :** En entrée cette fonctionnalité prendra un ensemble d'images composant la nouvelle classe à insérer dans la base. Les images seront pré-traitées par la fonctionnalité de pré-traitement des images afin de maintenir une homogénéité de la base. La fonctionnalité ne comporte aucune sortie.

— **Interactions :** Cette fonctionnalité va interagir avec la fonctionnalité de pré-traitement des images et avec la base d'apprentissage.
— **Traitement associé :** Pour chaque image en entrée, un pré-traitement sera effectué puis l'image traitée sauvegardée dans la base d'apprentissage.
— **Gestion des erreurs :** La gestion des erreurs sera faite par renvoi d'exceptions.

2.2 Spécifications non fonctionnelles

2.2.1 Contraintes de développement et conception

Langages de programmation

Le système sera à développer avec le langage C# et le *framework* .NET, dans ses versions 6 et 4.6+ respectivement.

Environnement et Logiciels

Le système à développer repose sur les outils suivants :

— Distribution Linux Debian 8 (jessie) 64 bits
— Serveur *RabbitMQ* (3.6.5)
— *Docker* (version 1.11.1, build 5604cbe)
— Système de *versionning Git*
— *Mono* et *MonoDevelop*, qui seront requis afin de permettre la bonne exécution du *framework* .NET sous Linux.

Contraintes de développement

Le développement du système devra se dérouler en 3 étapes :

1. : Reconnaissance d'un document à partir d'une image
2. : Lecture et analyse d'un document par OCR
3. : Analyse et recherche sémantique des données extraites par l'outil d'OCR

2.2.2 Contraintes de fonctionnement et d'exploitation

Performances

D'un point de vue utilisateur, l'application devra être la plus réactive possible. Dans la mesure où le traitement d'images peut devenir coûteux en temps (les images scannées sont généralement de très grande taille), un temps de latence est toujours à prévoir. Le but sera donc de diminuer ce temps de traitement au maximum.

3 Procédure d'installation de Mono/MonoDevelop (Debian 8 jessie)

3.1 Installation de Mono

- sudo apt-key adv –keyserver hkp ://keyserver.ubuntu.com :80 –recv-keys 3FA7E0328081BF F6A14DA29AA6A19B38D3D831EF
- echo "deb http ://download.mono-project.com/repo/debian wheezy main" | sudo tee /etc/apt/sources.list.d/mono-xamarin.list
- echo "deb http ://download.mono-project.com/repo/debian wheezy-libjpeg62-compat main" | sudo tee -a /etc/apt/sources.list.d/mono-xamarin.list
- sudo apt-get update
- sudo apt-get install mono-devel mono-complete referenceassemblies-pcl
- sudo apt-get install ca-certificates-mono [si support SSL et HTTPS necessaire]
- sudo apt-get install mono-xsp4 [necessaire pour compatibilité ASP.NET]

3.2 Installation de backports

- echo "deb http ://ftp.debian.org/debian jessie-backports main" | sudo tee -a /etc/apt/-sources.list.d/backports.list
- sudo apt-get update

3.3 Installation de flatpak

- apt-get -t jessie-backports install flatpak

3.4 Installation de MonoDevelop

- wget https ://sdk.gnome.org/keys/gnome-sdk.gpg
- flatpak remote-add –gpg-import=gnome-sdk.gpg gnome https ://sdk.gnome.org/repo/
- flatpak install gnome org.freedesktop.Platform 1.4
- cd [..]/Telechargements
- flatpak install –user –bundle monodevelop-6.1.2.44-1.flatpak

3.5 Installation des add-on packages

- nuint :
  - apt-get install monodevelop-nunit
- database
  - apt-get install monodevelop-database
- Git/Svn
  - echo "deb http ://security.debian.org/debian-security wheezy/updates main " | sudo tee -a /etc/apt/sources.list.d/monoversioncontrol.list
  - apt-get install monodevelop-versioncontrol

3.6 Installation de GtkSharp3

— apt-get install gtk-sharp3

3.7 Installation de cmake

— apt-get install cmake

3.8 Installation de libmono

— apt-get install libmono-cairo2.0-cil

3.9 Directives MonoDevelop

— Choisir .NetFramework 4.6+
— Options du projet
    — Executer
        — Général
            — Décocher "Exécuter sur une console externe"

4 Planning prévisionnel

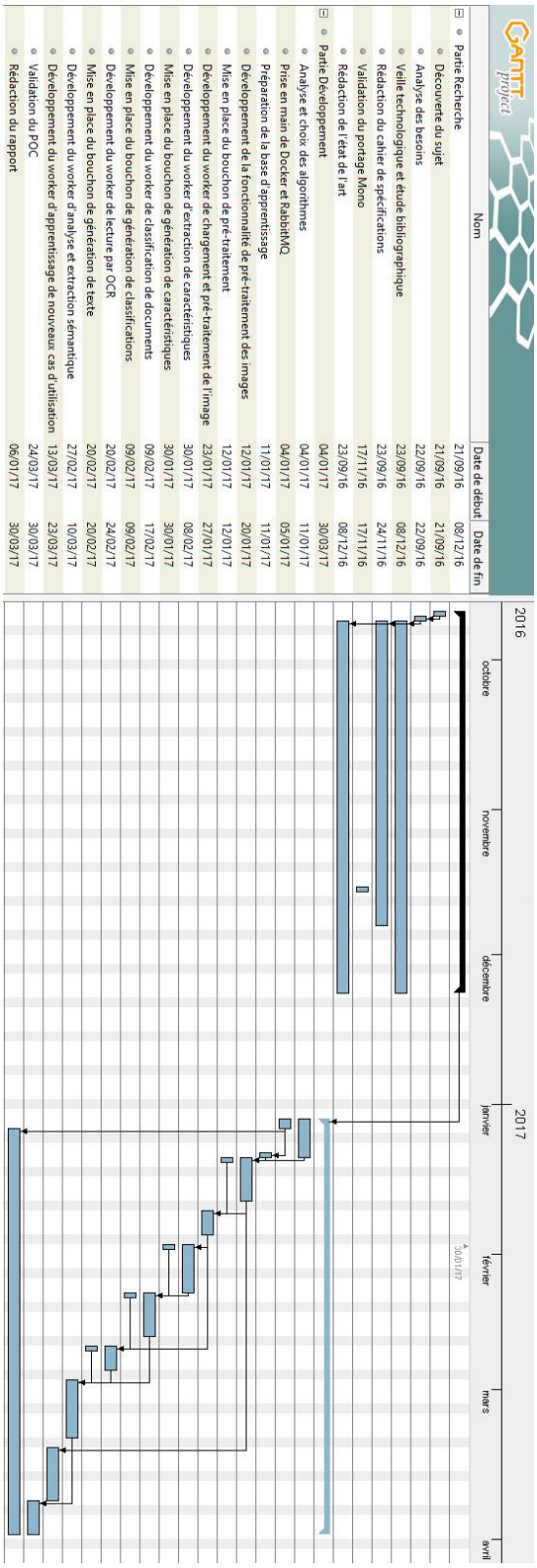


Figure 2 – *Planning prévisionnel du projet*

5 Planning réel

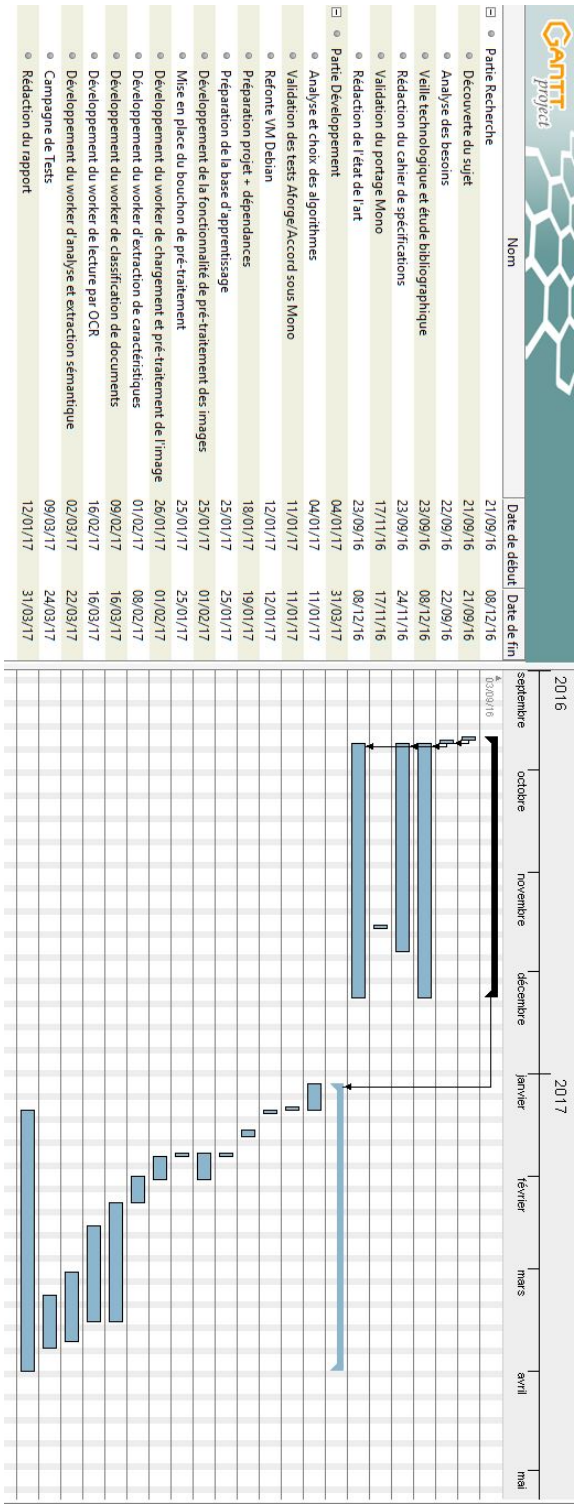


Figure 3 – *Planning réel du projet*

6 Structure globale du système

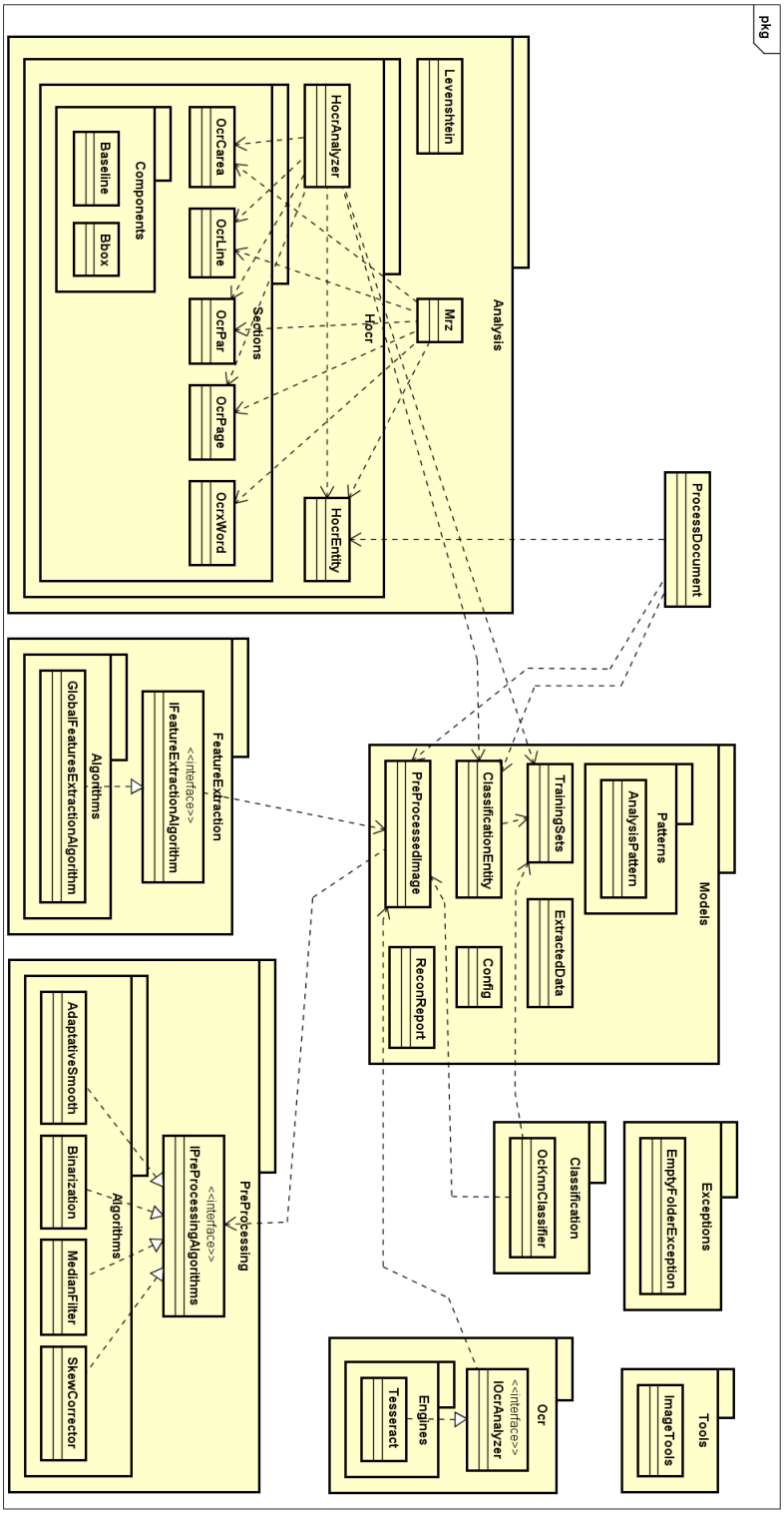


Figure 4 – *Structure globale du système*

Septième partie

Documentation

Reconnaissance d'éléments dans les documents juridiques

1.0

Generated by Doxygen 1.8.13

Contents

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

and the score obtained after the classification step 22

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 reconlib Namespace Reference

Namespaces

- namespace **Analysis**
- namespace **Classification**
- namespace **Exceptions**
- namespace **FeatureExtraction**
- namespace **Models**
- namespace **Ocr**
- namespace **PreProcessing**
- namespace **Tools**

Classes

- class **ProcessDocument**

  System's entry point.

5.2 reconlib.Analysis Namespace Reference

Namespaces

- namespace **Hocr**

Classes

- class **Levenshtein**

  Levenshtein distance. Computes the distance between two words.
- class **Mrz**

  Represents an MRZ (machine readeable zone)

5.3 reconlib.Analysis.Hocr Namespace Reference

Namespaces

- namespace **Sections**

Classes

- class **HocrAnalyzer**

  Analysis (p. 9) *and extraction of data from an hocr data structure*
- class **HocrEntity**

  Represents an hocr data structure

5.4 reconlib.Analysis.Hocr.Sections Namespace Reference

Namespaces

- namespace **Components**

Classes

- class **OcrCarea**

  Represents an hocr content area
- class **OcrLine**

  Represents an hocr line
- class **OcrPage**

  Represents an hocr page structure
- class **OcrPar**

  Represents an hocr paragraph
- class **OcrxWord**

  Represents an hocr word

5.5 reconlib.Analysis.Hocr.Sections.Components Namespace Reference

Classes

- class **Baseline**

  Represents an hocr baseline.
- class **Bbox**

  Represents an hocr bounding box

5.6 reconlib.Classification Namespace Reference

Classes

- class **OcKnnClassifier**

  OcKnn classifier wrapper

5.7 reconlib.Exceptions Namespace Reference

Classes

- class **EmptyFolderException**

  Exception meant to be thrown on empty folder

5.8 reconlib.FeatureExtraction Namespace Reference

Namespaces

- namespace **Algorithms**

Classes

- interface **IFeatureExtractionAlgorithm**

  Feature extraction algorithms interface

5.9 reconlib.FeatureExtraction.Algorithms Namespace Reference

Classes

- class **GlobalFeaturesExtractionAlgorithm**

  Global features extraction wrapper Extracts the global features of an image

5.10 reconlib.Models Namespace Reference

Namespaces

- namespace **Patterns**

Classes

- class **ClassificationEntity**

  ClassificationEntity *(*p. *22) represents the result data of a classification. This entity saves the*

See also

TrainingSet *(p. 65)*

and the score obtained after the classification step

- class **Config**

  Represents the xml config root node
- class **Configuration**

  Global access configuration data
- class **ExtractedData**

  ExtractedData *(p. 26) represents the data extracted after the semantic analysis step It allows to save the extracted content and its confidence*
- class **PreProcessedImage**

  Represents a processed image (pre-processing filters applied)
- class **ReconReport**

  Represents the report containing all the information from the system's execution
- class **Set**

  Represents an xml config set node
- class **Training**

  Global access configuration data
- class **TrainingSet**

  Represents a single training set xml node
- class **TrainingSets**

  Represents the training sets xml root node

5.11 reconlib.Models.Patterns Namespace Reference

Classes

- class **AnalysisPattern**

  Represents an analysis pattern. Each document must have it's own pattern
- class **Field**

  Represents a pattern field xml node
- class **Pattern**

  Represents the pattern xml root node

5.12 reconlib.Ocr Namespace Reference

Namespaces

- namespace **Engines**

Classes

- interface **IOcrAnalyzer**

  Ocr *(p. 12) wrappers interface*

5.13 reconlib.Ocr.Engines Namespace Reference

Classes

- class **Tesseract**

  Tesseract *(*p. *63) engine wrapper*

5.14 reconlib.PreProcessing Namespace Reference

Namespaces

- namespace **Algorithms**

Classes

- interface **IPreProcessingAlgorithm**

  Image pre-processing algorithms interface

5.15 reconlib.PreProcessing.Algorithms Namespace Reference

Classes

- class **AdaptiveSmooth**

  Adaptative smooth filter
- class **Binarization**

  Binarization *(*p. *20) filter.*
- class **MedianFilter**

  Median filter
- class **SkewCorrector**

  Image skew correction algorithm

5.16 reconlib.Tools Namespace Reference

Classes

- class **ImageTools**

  Image handling tools

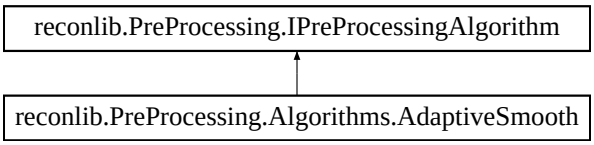
Chapter 6

Class Documentation

6.1 reconlib.PreProcessing.Algorithms.AdaptiveSmooth Class Reference

Adaptative smooth filter

Inheritance diagram for reconlib.PreProcessing.Algorithms.AdaptiveSmooth:



Public Member Functions

- Bitmap **Apply** (Bitmap image)
  Applies the filter to a Bitmap

Static Public Member Functions

- static **AdaptiveSmooth CreateNew** ()
  Class builder
- static **AdaptiveSmooth CreateNew** (double factor)
  Class builder

6.1.1 Detailed Description

Adaptative smooth filter

Definition at line 10 of file AdaptiveSmooth.cs.

6.1.2 Member Function Documentation

6.1.2.1 Apply()

```
Bitmap reconlib.PreProcessing.Algorithms.AdaptiveSmooth.Apply (
            Bitmap image )
```

Applies the filter to a Bitmap

Parameters

image	The bitmap to process

Returns

The processed bitmap

Implements **reconlib.PreProcessing.IPreProcessingAlgorithm** (p. 36).

Definition at line 46 of file AdaptiveSmooth.cs.

6.1.2.2 CreateNew() [1/2]

```
static AdaptiveSmooth reconlib.PreProcessing.Algorithms.AdaptiveSmooth.CreateNew ( ) [static]
```

Class builder

Returns

A new instance of AdaptativeSmooth filter

Definition at line 26 of file AdaptiveSmooth.cs.

6.1.2.3 CreateNew() [2/2]

```
static AdaptiveSmooth reconlib.PreProcessing.Algorithms.AdaptiveSmooth.CreateNew (
            double factor ) [static]
```

Class builder

Parameters

factor	The smoothing factor

Returns

A new instance of AdaptativeSmooth filter

Definition at line 36 of file AdaptiveSmooth.cs.

The documentation for this class was generated from the following file:

- D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/PreProcessing/↩ Algorithms/ **AdaptiveSmooth.cs**

6.2 reconlib.Analysis.Hocr.Sections.Components.Baseline Class Reference

Represents an hocr baseline.

Static Public Member Functions

- static **Baseline** **CreateNew** (double slope, double constant)
  Class builder

Properties

- double **Slope** `[get, set]`
  The baseline equation slope
- double **Constant** `[get, set]`
  The baseline equation constant

6.2.1 Detailed Description

Represents an hocr baseline.

Baseline (p. 17) 0.019 -22 would stand for y = 0.019 x - 22

Definition at line 10 of file Baseline.cs.

6.2.2 Member Function Documentation

6.2.2.1 CreateNew()

```
static  Baseline reconlib.Analysis.Hocr.Sections.Components.Baseline.CreateNew (
            double slope,
            double constant )  [static]
```

Class builder

Parameters

slope	The baseline slope
constant	The baseline constant

Returns

A new instance of **Baseline** (p. 17)

Definition at line 42 of file Baseline.cs.

6.2.3 Property Documentation

6.2.3.1 Constant

```
double reconlib.Analysis.Hocr.Sections.Components.Baseline.Constant  [get], [set]
```

The baseline equation constant

Definition at line 21 of file Baseline.cs.

6.2.3.2 Slope

```
double reconlib.Analysis.Hocr.Sections.Components.Baseline.Slope  [get], [set]
```

The baseline equation slope

Definition at line 16 of file Baseline.cs.

The documentation for this class was generated from the following file:

- D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Analysis/↩ Hocr/Sections/Components/ **Baseline.cs**

6.3 reconlib.Analysis.Hocr.Sections.Components.Bbox Class Reference

Represents an hocr bounding box

Static Public Member Functions

- static **Bbox** **CreateNew** (int startx, int starty, int endx, int endy)
  Class builder

Properties

- int **StartX** `[get, set]`
  Bounding box start X coordinate
- int **StartY** `[get, set]`
  Bounding box start Y coordinate
- int **EndX** `[get, set]`
  Bounding box end X coordinate
- int **EndY** `[get, set]`
  Bounding box end Y coordinate

6.3.1 Detailed Description

Represents an hocr bounding box

Definition at line 7 of file Bbox.cs.

6.3.2 Member Function Documentation

6.3.2.1 CreateNew()

```
static Bbox reconlib.Analysis.Hocr.Sections.Components.Bbox.CreateNew (
            int startx,
            int starty,
            int endx,
            int endy )  [static]
```

Class builder

Parameters

startx	The box start x coordinate
starty	The box start y coordinate
endx	The box end x coordinate
endy	The box end y coordinate

Returns

A new instance of **Bbox** (p. 18)

Definition at line 53 of file Bbox.cs.

6.3.3 Property Documentation

6.3.3.1 EndX

```
int reconlib.Analysis.Hocr.Sections.Components.Bbox.EndX  [get], [set]
```

Bounding box end X coordinate

Definition at line 22 of file Bbox.cs.

6.3.3.2 EndY

```
int reconlib.Analysis.Hocr.Sections.Components.Bbox.EndY  [get], [set]
```

Bounding box end Y coordinate

Definition at line 27 of file Bbox.cs.

6.3.3.3 StartX

```
int reconlib.Analysis.Hocr.Sections.Components.Bbox.StartX  [get], [set]
```

Bounding box start X coordinate

Definition at line 12 of file Bbox.cs.

6.3.3.4 StartY

```
int reconlib.Analysis.Hocr.Sections.Components.Bbox.StartY  [get], [set]
```

Bounding box start Y coordinate

Definition at line 17 of file Bbox.cs.

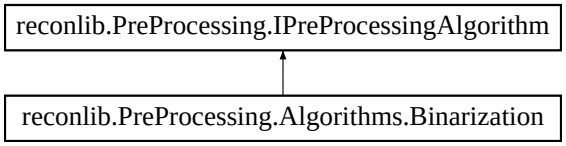
The documentation for this class was generated from the following file:

- D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Analysis/↩ Hocr/Sections/Components/ **Bbox.cs**

6.4 reconlib.PreProcessing.Algorithms.Binarization Class Reference

Binarization (p. 20) filter.

Inheritance diagram for reconlib.PreProcessing.Algorithms.Binarization:



Public Member Functions

- Bitmap **Apply** (Bitmap image)

  Applies the filter to a Bitmap

Static Public Member Functions

- static **Binarization** **CreateNew** ()
  Class builder

6.4.1 Detailed Description

Binarization (p. 20) filter.

Definition at line 9 of file Binarization.cs.

6.4.2 Member Function Documentation

6.4.2.1 Apply()

```
Bitmap reconlib.PreProcessing.Algorithms.Binarization.Apply (
            Bitmap image )
```

Applies the filter to a Bitmap

Parameters

image	The bitmap to process

Returns

The processed bitmap

Implements **reconlib.PreProcessing.IPreProcessingAlgorithm** (p. 36).

Definition at line 37 of file Binarization.cs.

6.4.2.2 CreateNew()

```
static Binarization reconlib.PreProcessing.Algorithms.Binarization.CreateNew ( ) [static]
```

Class builder

Returns

A new instance of **Binarization** (p. 20)

Definition at line 27 of file Binarization.cs.

The documentation for this class was generated from the following file:

- D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/PreProcessing/↩ Algorithms/ **Binarization.cs**

6.5 reconlib.Models.ClassificationEntity Class Reference

ClassificationEntity (p. 22) represents the result data of a classification. This entity saves the

See also

TrainingSet (p. 65)

and the score obtained after the classification step

Static Public Member Functions

- static **ClassificationEntity** **CreateNew** (**TrainingSet** selectedTrainingSet, double score)
  Class builder

Properties

- **TrainingSet** **SelectedTrainingSet** [get]
  The training set retained from the classification
- double **Score** [get]
  The score obtained

6.5.1 Detailed Description

ClassificationEntity (p. 22) represents the result data of a classification. This entity saves the

See also

TrainingSet (p. 65)

and the score obtained after the classification step

Definition at line 7 of file ClassificationEntity.cs.

6.5.2 Member Function Documentation

6.5.2.1 CreateNew()

```
static  ClassificationEntity reconlib.Models.ClassificationEntity.CreateNew (
             TrainingSet selectedTrainingSet,
            double score )  [static]
```

Class builder

Parameters

selectedTrainingSet	The corresponding training set
score	The classification score

Returns

A new **ClassificationEntity** (p. 22) instance

Definition at line 38 of file ClassificationEntity.cs.

6.5.3 Property Documentation

6.5.3.1 Score

```
double reconlib.Models.ClassificationEntity.Score  [get]
```

The score obtained

Definition at line 18 of file ClassificationEntity.cs.

6.5.3.2 SelectedTrainingSet

```
TrainingSet reconlib.Models.ClassificationEntity.SelectedTrainingSet  [get]
```

The training set retained from the classification

Definition at line 13 of file ClassificationEntity.cs.

The documentation for this class was generated from the following file:

- D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Models/ **ClassificationEntity.cs**

6.6 reconlib.Models.Config Class Reference

Represents the xml config root node

Properties

- List< **Set** > **Set** `[get, set]`

  Loaded configuration sets

6.6.1 Detailed Description

Represents the xml config root node

Definition at line 77 of file Config.cs.

6.6.2 Property Documentation

6.6.2.1 Set

```
List< Set> reconlib.Models.Config.Set  [get], [set]
```

Loaded configuration sets

Definition at line 83 of file Config.cs.

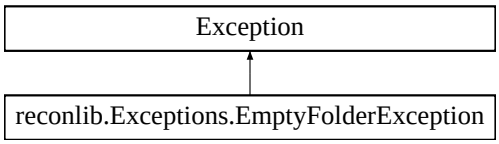
The documentation for this class was generated from the following file:

- D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Models/ **Config.cs**

6.7 reconlib.Exceptions.EmptyFolderException Class Reference

Exception meant to be thrown on empty folder

Inheritance diagram for reconlib.Exceptions.EmptyFolderException:



Public Member Functions

- **EmptyFolderException** ()

  Creates a new exception
- **EmptyFolderException** (string message)

  Creates a new exception and sets it's default message
- **EmptyFolderException** (string message, Exception inner)

  Creates a new exception and sets it's default message and the trigger exception

6.7.1 Detailed Description

Exception meant to be thrown on empty folder

Definition at line 8 of file EmptyFolderException.cs.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 EmptyFolderException() [1/3]

```
reconlib.Exceptions.EmptyFolderException.EmptyFolderException ( )
```

Creates a new exception

Definition at line 13 of file EmptyFolderException.cs.

6.7.2.2 EmptyFolderException() [2/3]

```
reconlib.Exceptions.EmptyFolderException.EmptyFolderException (
            string message )
```

Creates a new exception and sets it's default message

Parameters

message	The message to join to the exception

Definition at line 19 of file EmptyFolderException.cs.

6.7.2.3 EmptyFolderException() [3/3]

```
reconlib.Exceptions.EmptyFolderException.EmptyFolderException (
            string message,
            Exception inner )
```

Creates a new exception and sets it's default message and the trigger exception

Parameters

message	The message to join to the exception
inner	The exception that triggers this exception

Definition at line 26 of file EmptyFolderException.cs.

The documentation for this class was generated from the following file:

- D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Exceptions/**EmptyFolderException.cs**

6.8 reconlib.Models.ExtractedData Class Reference

ExtractedData (p. 26) represents the data extracted after the semantic analysis step It allows to save the extracted content and its confidence

Static Public Member Functions

- static **ExtractedData** **CreateNew** (string content, double confidence)
  Class builder

Properties

- string **Content** `[get, set]`
  The extracted data
- double **Confidence** `[get, set]`
  The data confidence (%)

6.8.1 Detailed Description

ExtractedData (p. 26) represents the data extracted after the semantic analysis step It allows to save the extracted content and its confidence

Definition at line 9 of file ExtractedData.cs.

6.8.2 Member Function Documentation

6.8.2.1 CreateNew()

```
static  ExtractedData reconlib.Models.ExtractedData.CreateNew (
            string content,
            double confidence )  [static]
```

Class builder

Parameters

content	The extracted content (string)
confidence	The average confidence of the content

Returns

A new instance of **ExtractedData** (p. 26)

Definition at line 42 of file ExtractedData.cs.

6.8.3 Property Documentation

6.8.3.1 Confidence

```
double reconlib.Models.ExtractedData.Confidence  [get], [set]
```

The data confidence (%)

Definition at line 20 of file ExtractedData.cs.

6.8.3.2 Content

```
string reconlib.Models.ExtractedData.Content  [get], [set]
```

The extracted data

Definition at line 15 of file ExtractedData.cs.

The documentation for this class was generated from the following file:

- D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Models/ **ExtractedData.cs**

6.9 reconlib.Models.Patterns.Field Class Reference

Represents a pattern field xml node

Properties

- string **Name** `[get, set]`

  The pattern field name. Must be unique
- string **Label** `[get, set]`

  The pattern field label. Used to detect the data to extract on the document
- string **Expected** `[get, set]`

  Number of words to extract after the label. Can be 1 ou +
- int **Delta** `[get, set]`

  Levenshtein distance delta.

6.9.1 Detailed Description

Represents a pattern field xml node

Definition at line 33 of file AnalysisPattern.cs.

6.9.2 Property Documentation

6.9.2.1 Delta

```
int reconlib.Models.Patterns.Field.Delta  [get], [set]
```

Levenshtein distance delta.

Definition at line 57 of file AnalysisPattern.cs.

6.9.2.2 Expected

```
string reconlib.Models.Patterns.Field.Expected  [get], [set]
```

Number of words to extract after the label. Can be 1 ou +

Definition at line 51 of file AnalysisPattern.cs.

6.9.2.3 Label

```
string reconlib.Models.Patterns.Field.Label  [get], [set]
```

The pattern field label. Used to detect the data to extract on the document

Definition at line 45 of file AnalysisPattern.cs.

6.9.2.4 Name

```
string reconlib.Models.Patterns.Field.Name  [get], [set]
```

The pattern field name. Must be unique

Definition at line 39 of file AnalysisPattern.cs.

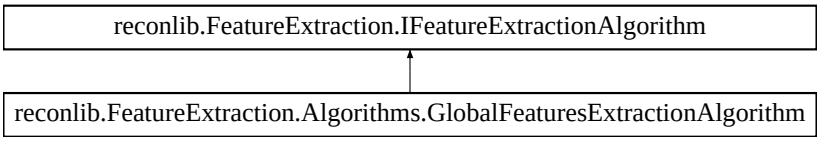
The documentation for this class was generated from the following file:

- D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Models/↩ Patterns/ **AnalysisPattern.cs**

6.10 reconlib.FeatureExtraction.Algorithms.GlobalFeaturesExtractionAlgorithm Class Reference

Global features extraction wrapper Extracts the global features of an image

Inheritance diagram for reconlib.FeatureExtraction.Algorithms.GlobalFeaturesExtractionAlgorithm:



Public Member Functions

- GlobalFeatures **Run** (**PreProcessedImage** image)
  Runs the global features extraction algorithm on an image

Static Public Member Functions

- static **GlobalFeaturesExtractionAlgorithm** **CreateNew** (string paramsFileUrl)
  *Creates a new instance of **GlobalFeaturesExtractionAlgorithm** (p. 29)*

6.10.1 Detailed Description

Global features extraction wrapper Extracts the global features of an image

Definition at line 11 of file GlobalFeaturesExtractionAlgorithm.cs.

6.10.2 Member Function Documentation

6.10.2.1 CreateNew()

```
static  GlobalFeaturesExtractionAlgorithm reconlib.FeatureExtraction.Algorithms.GlobalFeatures↩
ExtractionAlgorithm.CreateNew (
            string paramsFileUrl )  [static]
```

Creates a new instance of **GlobalFeaturesExtractionAlgorithm** (p. 29)

Parameters

paramsFileUrl	The global feature parameters file path

Returns

An instance of **GlobalFeaturesExtractionAlgorithm** (p. 29)

Definition at line 29 of file GlobalFeaturesExtractionAlgorithm.cs.

6.10.2.2 Run()

```
GlobalFeatures reconlib.FeatureExtraction.Algorithms.GlobalFeaturesExtractionAlgorithm.Run (
            PreProcessedImage image )
```

Runs the global features extraction algorithm on an image

Parameters

image	The image to analyse

Returns

An instance of GlobalFeatures containing all the features extracted from the image

Exceptions

FileNotFoundException	The parameters file was not found

Implements **reconlib.FeatureExtraction.IFeatureExtractionAlgorithm** (p. 34).

Definition at line 40 of file GlobalFeaturesExtractionAlgorithm.cs.

The documentation for this class was generated from the following file:

- D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Feature↩ Extraction/Algorithms/ **GlobalFeaturesExtractionAlgorithm.cs**

6.11 reconlib.Analysis.Hocr.HocrAnalyzer Class Reference

Analysis (p. 9) and extraction of data from an hocr data structure

Public Member Functions

- **Mrz ExtractMrz** (**HocrEntity** hocr)
  Extracts the MRZ if it exists on the document
- Dictionary< string, **ExtractedData** > **ExtractData** (**HocrEntity** hocr, **ClassificationEntity** classification)
  Extracts all the data from the document. The extraction is guided by the document classification

Static Public Member Functions

- static **HocrAnalyzer CreateNew** ()
  Static builder

6.11.1 Detailed Description

Analysis (p. 9) and extraction of data from an hocr data structure

Definition at line 14 of file HocrAnalyzer.cs.

6.11.2 Member Function Documentation

6.11.2.1 CreateNew()

```
static HocrAnalyzer reconlib.Analysis.Hocr.HocrAnalyzer.CreateNew ( ) [static]
```

Static builder

Returns

A new instance of **HocrAnalyzer** (p. 30)

Definition at line 25 of file HocrAnalyzer.cs.

6.11.2.2 ExtractData()

```
Dictionary<string, ExtractedData> reconlib.Analysis.Hocr.HocrAnalyzer.ExtractData (
            HocrEntity hocr,
            ClassificationEntity classification )
```

Extracts all the data from the document. The extraction is guided by the document classification

Parameters

hocr	The document's hocr extracted by the ocr engine Tesseract
classification	The document's classification ClassificationEntity

Returns

A dictionnary containing all the extracted data

Definition at line 46 of file HocrAnalyzer.cs.

6.11.2.3 ExtractMrz()

```
Mrz reconlib.Analysis.Hocr.HocrAnalyzer.ExtractMrz (
            HocrEntity hocr )
```

Extracts the MRZ if it exists on the document

Parameters

hocr	The document's hocr extracted by the ocr engine

Returns

An instance of **Mrz** (p. 39) containing the extracted data OR null if the MRZ doesn't exist on the document

Definition at line 35 of file HocrAnalyzer.cs.

The documentation for this class was generated from the following file:

- D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Analysis/Hocr/**HocrAnalyzer.cs**

6.12 reconlib.Analysis.Hocr.HocrEntity Class Reference

Represents an hocr data structure

Static Public Member Functions

- static **HocrEntity** **CreateNew** (string hocrData)

  Class builder

Properties

- List< **OcrPage** > **Pages** [get, set]

  Hocr *(p. 10) pages*
- string **Hocr** [get, set]

  Full hocr data in string format

6.12.1 Detailed Description

Represents an hocr data structure

Definition at line 14 of file HocrEntity.cs.

6.12.2 Member Function Documentation

6.12.2.1 CreateNew()

```
static HocrEntity reconlib.Analysis.Hocr.HocrEntity.CreateNew (
            string hocrData ) [static]
```

Class builder

Parameters

hocrData	Extracted data from the ocr engine

Returns

A new instance of **HocrEntity** (p. 32)

Definition at line 43 of file HocrEntity.cs.

6.12.3 Property Documentation

6.12.3.1 Hocr

```
string reconlib.Analysis.Hocr.HocrEntity.Hocr [get], [set]
```

Full hocr data in string format

Definition at line 25 of file HocrEntity.cs.

6.12.3.2 Pages

```
List< OcrPage> reconlib.Analysis.Hocr.HocrEntity.Pages [get], [set]
```

Hocr (p. 10) pages

Definition at line 20 of file HocrEntity.cs.

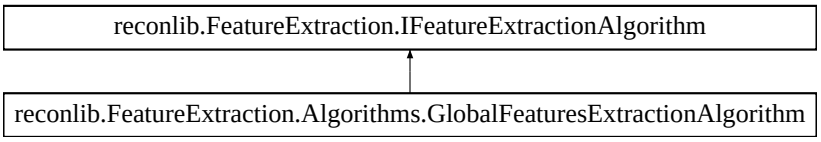
The documentation for this class was generated from the following file:

- D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Analysis/Hocr/**HocrEntity.cs**

6.13 reconlib.FeatureExtraction.IFeatureExtractionAlgorithm Interface Reference

Feature extraction algorithms interface

Inheritance diagram for reconlib.FeatureExtraction.IFeatureExtractionAlgorithm:



Public Member Functions

- GlobalFeatures **Run** (**PreProcessedImage** image)
  Runs the feature extraction algorithm on a given image

6.13.1 Detailed Description

Feature extraction algorithms interface

Definition at line 9 of file IFeatureExtractionAlgorithm.cs.

6.13.2 Member Function Documentation

6.13.2.1 Run()

```
GlobalFeatures reconlib.FeatureExtraction.IFeatureExtractionAlgorithm.Run (
              PreProcessedImage image )
```

Runs the feature extraction algorithm on a given image

Parameters

image	The image to process

Returns

AThe global features extracted

Implemented in **reconlib.FeatureExtraction.Algorithms.GlobalFeaturesExtractionAlgorithm** (p. 30).

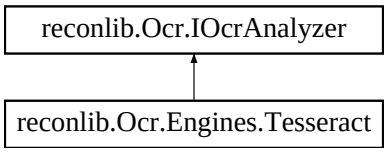
The documentation for this interface was generated from the following file:

- D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Feature↩ Extraction/ **IFeatureExtractionAlgorithm.cs**

6.14 reconlib.Ocr.IOcrAnalyzer Interface Reference

Ocr (p. 12) wrappers interface

Inheritance diagram for reconlib.Ocr.IOcrAnalyzer:



Public Member Functions

- string **Run** (**PreProcessedImage** imageToAnalyse)
  Runs the ocr engine on a given image

6.14.1 Detailed Description

Ocr (p. 12) wrappers interface

Definition at line 8 of file IOcrAnalyzer.cs.

6.14.2 Member Function Documentation

6.14.2.1 Run()

```
string reconlib.Ocr.IOcrAnalyzer.Run (
            PreProcessedImage imageToAnalyse )
```

Runs the ocr engine on a given image

Parameters

imageToAnalyse	The image to process

Returns

The extracted document content

Implemented in **reconlib.Ocr.Engines.Tesseract** (p. 64).

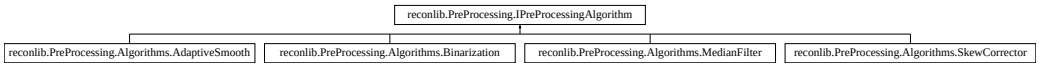
The documentation for this interface was generated from the following file:

- D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Ocr/**IOcr↩Analyzer.cs**

6.15 reconlib.PreProcessing.IPreProcessingAlgorithm Interface Reference

Image pre-processing algorithms interface

Inheritance diagram for reconlib.PreProcessing.IPreProcessingAlgorithm:



Public Member Functions

- Bitmap **Apply** (Bitmap image)

  Applies a filter/algorithm to a given image

6.15.1 Detailed Description

Image pre-processing algorithms interface

Definition at line 8 of file IPreProcessingAlgorithm.cs.

6.15.2 Member Function Documentation

6.15.2.1 Apply()

```
Bitmap reconlib.PreProcessing.IPreProcessingAlgorithm.Apply (
            Bitmap image )
```

Applies a filter/algorithm to a given image

Parameters

image	The image to process

Returns

The processed image

Implemented in **reconlib.PreProcessing.Algorithms.MedianFilter** (p. 37), **reconlib.PreProcessing.Algorithms.SkewCorrector** (p. 62), **reconlib.PreProcessing.Algorithms.AdaptiveSmooth** (p. 15), and **reconlib.PreProcessing.Algorithms.Binarization** (p. 21).

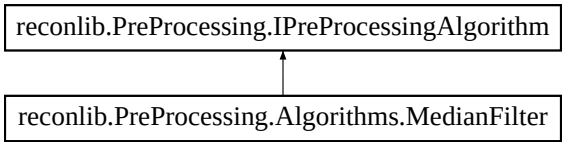
The documentation for this interface was generated from the following file:

- D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/PreProcessing/**IPreProcessingAlgorithm.cs**

6.16 reconlib.PreProcessing.Algorithms.MedianFilter Class Reference

Median filter

Inheritance diagram for reconlib.PreProcessing.Algorithms.MedianFilter:



Public Member Functions

- Bitmap **Apply** (Bitmap image)
  Applies the filter to a Bitmap

Static Public Member Functions

- static **MedianFilter CreateNew** ()
  Class builder
- static **MedianFilter CreateNew** (int windowSize)
  Class builder

Properties

- int **WindowSize** `[get]`
  The filter's window size

6.16.1 Detailed Description

Median filter

Definition at line 9 of file MedianFilter.cs.

6.16.2 Member Function Documentation

6.16.2.1 Apply()

```
Bitmap reconlib.PreProcessing.Algorithms.MedianFilter.Apply (
            Bitmap image )
```

Applies the filter to a Bitmap

Parameters

image	The bitmap to process

Returns

The processed bitmap

Implements **reconlib.PreProcessing.IPreProcessingAlgorithm** (p. 36).

Definition at line 56 of file MedianFilter.cs.

6.16.2.2 CreateNew() [1/2]

```
static MedianFilter reconlib.PreProcessing.Algorithms.MedianFilter.CreateNew ( ) [static]
```

Class builder

Returns

A new instance of **MedianFilter** (p. 37)

Definition at line 36 of file MedianFilter.cs.

6.16.2.3 CreateNew() [2/2]

```
static MedianFilter reconlib.PreProcessing.Algorithms.MedianFilter.CreateNew (
            int windowSize ) [static]
```

Class builder

Parameters

windowSize	The processing window's size

Returns

A new instance of **MedianFilter** (p. 37)

Definition at line 46 of file MedianFilter.cs.

6.16.3 Property Documentation

6.16.3.1 WindowSize

```
int reconlib.PreProcessing.Algorithms.MedianFilter.WindowSize  [get]
```

The filter's window size

Definition at line 17 of file MedianFilter.cs.

The documentation for this class was generated from the following file:

- D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/PreProcessing/↩ Algorithms/ **MedianFilter.cs**

6.17 reconlib.Analysis.Mrz Class Reference

Represents an MRZ (machine readeable zone)

Public Member Functions

- **Mrz** (string line1, string line2)
  Default contructor
- string **GetNationality** ()
  Extracts the person's nationality trigram from the mrz data
- string **GetLastName** ()
  Extracts the person's lastname from the mrz data
- string **GetFirstName** ()
  Extracts the person's firstname from the mrz data
- string **GetId** ()
  Extracts the document ID from the mrz data
- string **GetGender** ()
  Extracts the person's gender from the mrz data
- string **GetBirth** ()
  Extracts the person's birth date from the mrz data
- bool **IsValid** ()
  Verifies if the mrz data sequence is valid, by computing the required keys and verifying the data structure
- bool **IsStructureValid** ()
  Verifies if the mrz structure is valid
- bool **IsDataValid** ()
  Verifies if the mrz data is valid
- bool **IsIdValid** ()
  Verifies if the mrz ID is valid
- bool **IsBirthDateValid** ()
  Verifies if the mrz Birth Date is valid

Static Public Member Functions

- static int **ProcessControlKey** (string data)

  Calculates the control key of a CNI string
- static bool **HasMrz** (**HocrEntity** hocr)

  Verifies if a given hocr structure contains MRZ data
- static **Mrz** **ExtractMrz** (**HocrEntity** hocr)

  Extracts the MRZ data from an hocr document

Properties

- string **Line1** `[get]`

  MRZ first line
- string **Line2** `[get]`

  MRZ second line

6.17.1 Detailed Description

Represents an MRZ (machine readeable zone)

Definition at line 11 of file Mrz.cs.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 Mrz()

```
reconlib.Analysis.Mrz.Mrz (
            string line1,
            string line2 )
```

Default contructor

Parameters

line1	First mrz line
line2	Seconde mrz line

Definition at line 28 of file Mrz.cs.

6.17.3 Member Function Documentation

6.17.3.1 ExtractMrz()

```
static Mrz reconlib.Analysis.Mrz.ExtractMrz (
            HocrEntity hocr ) [static]
```

Extracts the MRZ data from an hocr document

Parameters

hocr	The hocr document to process

Returns

The MRZ instance if extracted, null otherwise

Definition at line 177 of file Mrz.cs.

6.17.3.2 GetBirth()

```
string reconlib.Analysis.Mrz.GetBirth ( )
```

Extracts the person's birth date from the mrz data

Returns

The extracted string

Definition at line 83 of file Mrz.cs.

6.17.3.3 GetFirstName()

```
string reconlib.Analysis.Mrz.GetFirstName ( )
```

Extracts the person's firstname from the mrz data

Returns

The extracted string

Definition at line 56 of file Mrz.cs.

6.17.3.4 GetGender()

```
string reconlib.Analysis.Mrz.GetGender ( )
```

Extracts the person's gender from the mrz data

Returns

The extracted string

Definition at line 74 of file Mrz.cs.

6.17.3.5 GetId()

```
string reconlib.Analysis.Mrz.GetId ( )
```

Extracts the document ID from the mrz data

Returns

The extracted ID

Definition at line 65 of file Mrz.cs.

6.17.3.6 GetLastName()

```
string reconlib.Analysis.Mrz.GetLastName ( )
```

Extracts the person's lastname from the mrz data

Returns

The extracted string

Definition at line 47 of file Mrz.cs.

6.17.3.7 GetNationality()

```
string reconlib.Analysis.Mrz.GetNationality ( )
```

Extracts the person's nationality trigram from the mrz data

Returns

The nationality trigram (ex: fra)

Definition at line 38 of file Mrz.cs.

6.17.3.8 HasMrz()

```
static bool reconlib.Analysis.Mrz.HasMrz (
              HocrEntity hocr )  [static]
```

Verifies if a given hocr structure contains MRZ data

Parameters

hocr	The hocr to process

Returns

True if the hocr has an MRZ, false otherwise

Definition at line 167 of file Mrz.cs.

6.17.3.9 IsBirthDateValid()

```
bool reconlib.Analysis.Mrz.IsBirthDateValid ( )
```

Verifies if the mrz Birth Date is valid

Returns

True if the date is valid, false otherwise

Definition at line 132 of file Mrz.cs.

6.17.3.10 IsDataValid()

```
bool reconlib.Analysis.Mrz.IsDataValid ( )
```

Verifies if the mrz data is valid

Returns

True if the data is valid, false otherwise

Definition at line 111 of file Mrz.cs.

6.17.3.11 IsIdValid()

```
bool reconlib.Analysis.Mrz.IsIdValid ( )
```

Verifies if the mrz ID is valid

Returns

True if the ID is valid, false otherwise

Definition at line 123 of file Mrz.cs.

6.17.3.12 IsStructureValid()

```
bool reconlib.Analysis.Mrz.IsStructureValid ( )
```

Verifies if the mrz structure is valid

Returns

True if it's a valid structure, false otherwise

Definition at line 101 of file Mrz.cs.

6.17.3.13 IsValid()

```
bool reconlib.Analysis.Mrz.IsValid ( )
```

Verifies if the mrz data sequence is valid, by computing the required keys and verifying the data structure

Returns

True if the mrz data sequence is valid, false otherwise

Definition at line 92 of file Mrz.cs.

6.17.3.14 ProcessControlKey()

```
static int reconlib.Analysis.Mrz.ProcessControlKey (
            string data )  [static]
```

Calculates the control key of a CNI string

Parameters

data	The string to hash

Returns

The control key

Definition at line 142 of file Mrz.cs.

6.17.4 Property Documentation

6.17.4.1 Line1

```
string reconlib.Analysis.Mrz.Line1  [get]
```

MRZ first line

Definition at line 16 of file Mrz.cs.

6.17.4.2 Line2

```
string reconlib.Analysis.Mrz.Line2  [get]
```

MRZ second line

Definition at line 21 of file Mrz.cs.

The documentation for this class was generated from the following file:

- D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Analysis/**Mrz.cs**

6.18 reconlib.Analysis.Hocr.Sections.OcrCarea Class Reference

Represents an hocr content area

Public Member Functions

- **OcrCarea** ()
  Default constructor

Properties

- string **Id** `[get, set]`
  The content area ID
- **Bbox** **Bbox** `[get, set]`
  The content area bounding box
- List< **OcrPar** > **Paragraphs** `[get, set]`
  The content area paragraphs

6.18.1 Detailed Description

Represents an hocr content area

Definition at line 10 of file OcrCarea.cs.

6.18.2 Constructor & Destructor Documentation

6.18.2.1 OcrCarea()

```
reconlib.Analysis.Hocr.Sections.OcrCarea.OcrCarea ( )
```

Default constructor

Definition at line 32 of file OcrCarea.cs.

6.18.3 Property Documentation

6.18.3.1 Bbox

```
Bbox reconlib.Analysis.Hocr.Sections.OcrCarea.Bbox  [get], [set]
```

The content area bounding box

Definition at line 21 of file OcrCarea.cs.

6.18.3.2 Id

```
string reconlib.Analysis.Hocr.Sections.OcrCarea.Id  [get], [set]
```

The content area ID

Definition at line 16 of file OcrCarea.cs.

6.18.3.3 Paragraphs

```
List< OcrPar> reconlib.Analysis.Hocr.Sections.OcrCarea.Paragraphs  [get], [set]
```

The content area paragraphs

Definition at line 26 of file OcrCarea.cs.

The documentation for this class was generated from the following file:

- D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Analysis/↩ Hocr/Sections/ **OcrCarea.cs**

6.19 reconlib.Analysis.Hocr.Sections.OcrLine Class Reference

Represents an hocr line

Public Member Functions

- **OcrLine** ()
  Default constructor

Properties

- string **Id** `[get, set]`
  The line ID
- **Bbox Bbox** `[get, set]`
  The line's bounding box (placement on the document)
- **Baseline Baseline** `[get, set]`
  Line's baseline orientation
- List< **OcrxWord** > **Words** `[get, set]`
  The line words

6.19.1 Detailed Description

Represents an hocr line

Definition at line 10 of file OcrLine.cs.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 OcrLine()

```
reconlib.Analysis.Hocr.Sections.OcrLine.OcrLine ( )
```

Default constructor

Definition at line 37 of file OcrLine.cs.

6.19.3 Property Documentation

6.19.3.1 Baseline

```
Baseline reconlib.Analysis.Hocr.Sections.OcrLine.Baseline [get], [set]
```

Line's baseline orientation

Definition at line 26 of file OcrLine.cs.

6.19.3.2 Bbox

```
Bbox reconlib.Analysis.Hocr.Sections.OcrLine.Bbox [get], [set]
```

The line's bounding box (placement on the document)

Definition at line 21 of file OcrLine.cs.

6.19.3.3 Id

```
string reconlib.Analysis.Hocr.Sections.OcrLine.Id [get], [set]
```

The line ID

Definition at line 16 of file OcrLine.cs.

6.19.3.4 Words

```
List< OcrxWord> reconlib.Analysis.Hocr.Sections.OcrLine.Words [get], [set]
```

The line words

Definition at line 31 of file OcrLine.cs.

The documentation for this class was generated from the following file:

- D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Analysis/↩ Hocr/Sections/ **OcrLine.cs**

6.20 reconlib.Analysis.Hocr.Sections.OcrPage Class Reference

Represents an hocr page structure

Public Member Functions

- **OcrPage** ()
  Default constructor

Properties

- string **Id** `[get, set]`
  The page ID
- string **Image** `[get, set]`
  The page image (if set)
- **Bbox** **Bbox** `[get, set]`
  The page bounding box (placement on the document)
- List< **OcrCarea** > **ContentAreas** `[get, set]`
  The page content areas
- int **Ppageno** `[get, set]`
  The page number

6.20.1 Detailed Description

Represents an hocr page structure

Definition at line 10 of file OcrPage.cs.

6.20.2 Constructor & Destructor Documentation

6.20.2.1 OcrPage()

```
reconlib.Analysis.Hocr.Sections.OcrPage.OcrPage ( )
```

Default constructor

Definition at line 43 of file OcrPage.cs.

6.20.3 Property Documentation

6.20.3.1 Bbox

```
Bbox reconlib.Analysis.Hocr.Sections.OcrPage.Bbox [get], [set]
```

The page bounding box (placement on the document)

Definition at line 26 of file OcrPage.cs.

6.20.3.2 ContentAreas

```
List< OcrCarea> reconlib.Analysis.Hocr.Sections.OcrPage.ContentAreas  [get], [set]
```

The page content areas

Definition at line 31 of file OcrPage.cs.

6.20.3.3 Id

```
string reconlib.Analysis.Hocr.Sections.OcrPage.Id  [get], [set]
```

The page ID

Definition at line 16 of file OcrPage.cs.

6.20.3.4 Image

```
string reconlib.Analysis.Hocr.Sections.OcrPage.Image  [get], [set]
```

The page image (if set)

Definition at line 21 of file OcrPage.cs.

6.20.3.5 Ppageno

```
int reconlib.Analysis.Hocr.Sections.OcrPage.Ppageno  [get], [set]
```

The page number

Definition at line 37 of file OcrPage.cs.

The documentation for this class was generated from the following file:

- D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Analysis/↩ Hocr/Sections/ **OcrPage.cs**

6.21 reconlib.Analysis.Hocr.Sections.OcrPar Class Reference

Represents an hocr paragraph

Public Member Functions

- **OcrPar** ()
  Default constructor

Properties

- string **Id** `[get, set]`
  The paragraph's ID
- **Bbox** **Bbox** `[get, set]`
  The paragraphs bounding box (placement on the document)
- List< **OcrLine** > **Lines** `[get, set]`
  The paragraph lines

6.21.1 Detailed Description

Represents an hocr paragraph

Definition at line 10 of file OcrPar.cs.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 OcrPar()

```
reconlib.Analysis.Hocr.Sections.OcrPar.OcrPar ( )
```

Default constructor

Definition at line 32 of file OcrPar.cs.

6.21.3 Property Documentation

6.21.3.1 Bbox

```
Bbox reconlib.Analysis.Hocr.Sections.OcrPar.Bbox  [get], [set]
```

The paragraphs bounding box (placement on the document)

Definition at line 21 of file OcrPar.cs.

6.21.3.2 Id

```
string reconlib.Analysis.Hocr.Sections.OcrPar.Id  [get], [set]
```

The paragraph's ID

Definition at line 16 of file OcrPar.cs.

6.21.3.3 Lines

```
List< OcrLine> reconlib.Analysis.Hocr.Sections.OcrPar.Lines  [get], [set]
```

The paragraph lines

Definition at line 26 of file OcrPar.cs.

The documentation for this class was generated from the following file:

- D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Analysis/↩ Hocr/Sections/ **OcrPar.cs**

6.22 reconlib.Analysis.Hocr.Sections.OcrxWord Class Reference

Represents an hocr word

Properties

- string **Id** `[get, set]`
  The word id
- **Bbox Bbox** `[get, set]`
  The word bounding box (placement on the document)
- int **Conf** `[get, set]`
  The word confidence (%)
- string **Lang** `[get, set]`
  Detected word language
- string **Content** `[get, set]`
  The word

6.22.1 Detailed Description

Represents an hocr word

Definition at line 9 of file OcrxWord.cs.

6.22.2 Property Documentation

6.22.2.1 Bbox

```
Bbox reconlib.Analysis.Hocr.Sections.OcrxWord.Bbox  [get], [set]
```

The word bounding box (placement on the document)

Definition at line 20 of file OcrxWord.cs.

6.22.2.2 Conf

```
int reconlib.Analysis.Hocr.Sections.OcrxWord.Conf  [get], [set]
```

The word confidence (%)

Definition at line 25 of file OcrxWord.cs.

6.22.2.3 Content

```
string reconlib.Analysis.Hocr.Sections.OcrxWord.Content  [get], [set]
```

The word

Definition at line 35 of file OcrxWord.cs.

6.22.2.4 Id

```
string reconlib.Analysis.Hocr.Sections.OcrxWord.Id  [get], [set]
```

The word id

Definition at line 15 of file OcrxWord.cs.

6.22.2.5 Lang

```
string reconlib.Analysis.Hocr.Sections.OcrxWord.Lang [get], [set]
```

Detected word language

Definition at line 30 of file OcrxWord.cs.

The documentation for this class was generated from the following file:

- D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Analysis/↩ Hocr/Sections/ **OcrxWord.cs**

6.23 reconlib.Models.Patterns.Pattern Class Reference

Represents the pattern xml root node

Properties

- List< **Field** > **Field** `[get, set]`
  List of the pattern fields

6.23.1 Detailed Description

Represents the pattern xml root node

Definition at line 64 of file AnalysisPattern.cs.

6.23.2 Property Documentation

6.23.2.1 Field

```
List< Field> reconlib.Models.Patterns.Pattern.Field [get], [set]
```

List of the pattern fields

Definition at line 70 of file AnalysisPattern.cs.

The documentation for this class was generated from the following file:

- D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Models/↩ Patterns/ **AnalysisPattern.cs**

6.24 reconlib.Models.PreProcessedImage Class Reference

Represents a processed image (pre-processing filters applied)

Public Types

- enum **Extensions** { **Extensions.Png**, **Extensions.Jpeg**, **Extensions.Tiff** }
  Compatible extensions for the save method

Public Member Functions

- void **SaveTo** (string url, **Extensions** ext)
  Saves the preprocessed image to disk

Static Public Member Functions

- static **PreProcessedImage** **ProcessNew** (Bitmap originalBitmap, **Binarization** binarizationFilter, params **IPreProcessingAlgorithm**[ ] preProcessingAlgorithms)
  Class builder

Properties

- Bitmap **OriginalImage** `[get]`
  The original image
- Bitmap **BinaryImage** `[get]`
  The binary image, after the binarization step
- Bitmap **ProcessedImage** `[get]`
  The preprocessed image, after all the preprocessing algorithms were applied

6.24.1 Detailed Description

Represents a processed image (pre-processing filters applied)

Definition at line 14 of file PreProcessedImage.cs.

6.24.2 Member Enumeration Documentation

6.24.2.1 Extensions

`enum` **reconlib.Models.PreProcessedImage.Extensions** `[strong]`

Compatible extensions for the save method

See also

PreProcessedImage.SaveTo (p. 56)

Enumerator

Png	
Jpeg	
Tiff	

Definition at line 37 of file PreProcessedImage.cs.

6.24.3 Member Function Documentation

6.24.3.1 ProcessNew()

```
static PreProcessedImage reconlib.Models.PreProcessedImage.ProcessNew (
            Bitmap originalBitmap,
             Binarization binarizationFilter,
            params IPreProcessingAlgorithm [] preProcessingAlgorithms ) [static]
```

Class builder

Parameters

originalBitmap	The image to process
binarizationFilter	The binarization filter to apply
preProcessingAlgorithms	The preprocessing algorithms to apply before processing the image

Returns

Creates a new instance of **PreProcessedImage** (p. 55)

Definition at line 83 of file PreProcessedImage.cs.

6.24.3.2 SaveTo()

```
void reconlib.Models.PreProcessedImage.SaveTo (
            string url,
             Extensions ext )
```

Saves the preprocessed image to disk

Parameters

url	Path were the image will be saved
ext	The image extension to be saved [TIFF by default]

Exceptions

Exception	An error occured during the save

Definition at line 94 of file PreProcessedImage.cs.

6.24.4 Property Documentation

6.24.4.1 BinaryImage

```
Bitmap reconlib.Models.PreProcessedImage.BinaryImage  [get]
```

The binary image, after the binarization step

Definition at line 25 of file PreProcessedImage.cs.

6.24.4.2 OriginalImage

```
Bitmap reconlib.Models.PreProcessedImage.OriginalImage  [get]
```

The original image

Definition at line 20 of file PreProcessedImage.cs.

6.24.4.3 ProcessedImage

```
Bitmap reconlib.Models.PreProcessedImage.ProcessedImage  [get]
```

The preprocessed image, after all the preprocessing algorithms were applied

Definition at line 30 of file PreProcessedImage.cs.

The documentation for this class was generated from the following file:

- D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Models/ **PreProcessedImage.cs**

6.25 reconlib.ProcessDocument Class Reference

System's entry point.

Public Member Functions

- **ReconReport Run** (string imageToProcessUrl)
  Runs the system on the selected image

Static Public Member Functions

- static **ProcessDocument Init** ()
  *Initializes and returns an instance of **ProcessDocument*** (p. *57*)

6.25.1 Detailed Description

System's entry point.

Definition at line 15 of file ProcessDocument.cs.

6.25.2 Member Function Documentation

6.25.2.1 Init()

```
static ProcessDocument reconlib.ProcessDocument.Init ( ) [static]
```

Initializes and returns an instance of **ProcessDocument** (p. 57)

Returns

An instance of **ProcessDocument** (p. 57)

Definition at line 72 of file ProcessDocument.cs.

6.25.2.2 Run()

```
ReconReport reconlib.ProcessDocument.Run (
            string imageToProcessUrl )
```

Runs the system on the selected image

Parameters

imageToProcessUrl	The image to process

Returns

A report of the executed tasks

Definition at line 82 of file ProcessDocument.cs.

The documentation for this class was generated from the following file:

- D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/**ProcessDocument.cs**

6.26 reconlib.Models.ReconReport Class Reference

Represents the report containing all the information from the system's execution

Properties

- **PreProcessedImage ProcessedImage** `[get, set]`
  The processed document image
- **HocrEntity Hocr** `[get, set]`
  The hocr data extracted from the document
- **ClassificationEntity Classification** `[get, set]`
  The document classification
- Dictionary< string, **ExtractedData** > **ExtractedData** `[get, set]`
  The data extracted from the hocr based on the documents classification

6.26.1 Detailed Description

Represents the report containing all the information from the system's execution

Definition at line 9 of file ReconReport.cs.

6.26.2 Property Documentation

6.26.2.1 Classification

ClassificationEntity reconlib.Models.ReconReport.Classification `[get], [set]`

The document classification

Definition at line 24 of file ReconReport.cs.

6.26.2.2 ExtractedData

```
Dictionary<string, ExtractedData> reconlib.Models.ReconReport.ExtractedData [get], [set]
```

The data extracted from the hocr based on the documents classification

Definition at line 29 of file ReconReport.cs.

6.26.2.3 Hocr

```
HocrEntity reconlib.Models.ReconReport.Hocr [get], [set]
```

The hocr data extracted from the document

Definition at line 19 of file ReconReport.cs.

6.26.2.4 ProcessedImage

```
PreProcessedImage reconlib.Models.ReconReport.ProcessedImage [get], [set]
```

The processed document image

Definition at line 14 of file ReconReport.cs.

The documentation for this class was generated from the following file:

- D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Models/ **ReconReport.cs**

6.27 reconlib.Models.Set Class Reference

Represents an xml config set node

Properties

- string **Key** [get, set]
  Configuration set key. Must be unique
- string **Value** [get, set]
  The configuration value

6.27.1 Detailed Description

Represents an xml config set node

Definition at line 58 of file Config.cs.

6.27.2 Property Documentation

6.27.2.1 Key

```
string reconlib.Models.Set.Key  [get], [set]
```

Configuration set key. Must be unique

Definition at line 64 of file Config.cs.

6.27.2.2 Value

```
string reconlib.Models.Set.Value  [get], [set]
```

The configuration value

Definition at line 70 of file Config.cs.

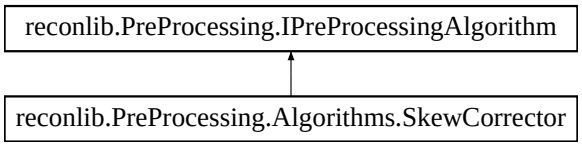
The documentation for this class was generated from the following file:

- D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Models/ **Config.cs**

6.28 reconlib.PreProcessing.Algorithms.SkewCorrector Class Reference

Image skew correction algorithm

Inheritance diagram for reconlib.PreProcessing.Algorithms.SkewCorrector:



Public Member Functions

- double **GetSkewAngle** (Bitmap image)

  Detects the skew angle of an image
- Bitmap **Apply** (Bitmap image)

  Applies the filter to a Bitmap

Static Public Member Functions

- static **SkewCorrector CreateNew** ()
  Class builder

6.28.1 Detailed Description

Image skew correction algorithm

Definition at line 10 of file SkewCorrector.cs.

6.28.2 Member Function Documentation

6.28.2.1 Apply()

```
Bitmap reconlib.PreProcessing.Algorithms.SkewCorrector.Apply (
            Bitmap image )
```

Applies the filter to a Bitmap

Parameters

image	The bitmap to process

Returns

The processed bitmap

Implements **reconlib.PreProcessing.IPreProcessingAlgorithm** (p. 36).

Definition at line 49 of file SkewCorrector.cs.

6.28.2.2 CreateNew()

```
static SkewCorrector reconlib.PreProcessing.Algorithms.SkewCorrector.CreateNew ( ) [static]
```

Class builder

Returns

A new instance of **SkewCorrector** (p. 61)

Definition at line 28 of file SkewCorrector.cs.

6.28.2.3 GetSkewAngle()

```
double reconlib.PreProcessing.Algorithms.SkewCorrector.GetSkewAngle (
            Bitmap image )
```

Detects the skew angle of an image

Parameters

image	The image to analyse

Returns

The angle value

Definition at line 38 of file SkewCorrector.cs.

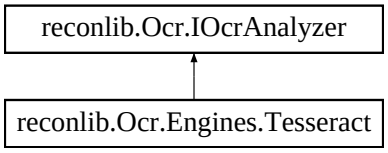
The documentation for this class was generated from the following file:

- D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/PreProcessing/↩ Algorithms/ **SkewCorrector.cs**

6.29 reconlib.Ocr.Engines.Tesseract Class Reference

Tesseract (p. 63) engine wrapper

Inheritance diagram for reconlib.Ocr.Engines.Tesseract:



Public Member Functions

- string **Run** (**PreProcessedImage** imageToAnalyse)

  *Runs the **Tesseract** (p. 63) engine on a selected image*

Static Public Member Functions

- static **Tesseract** **CreateNew** ()

  Class builder
- static EngineMode **GetEngineMode** (string config)

  Convert a configuration string engine mode

6.29.1 Detailed Description

Tesseract (p. 63) engine wrapper

Definition at line 10 of file Tesseract.cs.

6.29.2 Member Function Documentation

6.29.2.1 CreateNew()

```
static  Tesseract reconlib.Ocr.Engines.Tesseract.CreateNew ( )  [static]
```

Class builder

Returns

A new instance of **Tesseract** (p. 63)

Definition at line 23 of file Tesseract.cs.

6.29.2.2 GetEngineMode()

```
static EngineMode reconlib.Ocr.Engines.Tesseract.GetEngineMode (
            string config )  [static]
```

Convert a configuration string engine mode

Parameters

config	The string to interpret

Returns

The EngineMode associated to the given string

Definition at line 57 of file Tesseract.cs.

6.29.2.3 Run()

```
string reconlib.Ocr.Engines.Tesseract.Run (
            PreProcessedImage imageToAnalyse )
```

Runs the **Tesseract** (p. 63) engine on a selected image

Parameters

imageToAnalyse	The image to analyse

Returns

An hocr formatted string representing the contents of the image given

Implements **reconlib.Ocr.IOcrAnalyzer** (p. 35).

Definition at line 33 of file Tesseract.cs.

The documentation for this class was generated from the following file:

- D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Ocr/Engines/**Tesseract.cs**

6.30 reconlib.Models.TrainingSet Class Reference

Represents a single training set xml node

Properties

- string **Id** `[get, set]`
  The training set ID. Must be unique
- string **Path** `[get, set]`
  The training set directory path
- string **Pattern** `[get, set]`
  The syntax pattern associated to the training set

6.30.1 Detailed Description

Represents a single training set xml node

Definition at line 151 of file TrainingSets.cs.

6.30.2 Property Documentation

6.30.2.1 Id

```
string reconlib.Models.TrainingSet.Id  [get], [set]
```

The training set ID. Must be unique

Definition at line 157 of file TrainingSets.cs.

6.30.2.2 Path

`string reconlib.Models.TrainingSet.Path [get], [set]`

The training set directory path

Definition at line 163 of file TrainingSets.cs.

6.30.2.3 Pattern

`string reconlib.Models.TrainingSet.Pattern [get], [set]`

The syntax pattern associated to the training set

Definition at line 169 of file TrainingSets.cs.

The documentation for this class was generated from the following file:

- D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Models/ **TrainingSets.cs**

6.31 reconlib.Models.TrainingSets Class Reference

Represents the training sets xml root node

Properties

- List< **TrainingSet** > **TrainingSet** `[get, set]`
  The loaded training sets list

6.31.1 Detailed Description

Represents the training sets xml root node

Definition at line 176 of file TrainingSets.cs.

6.31.2 Property Documentation

6.31.2.1 TrainingSet

`List< TrainingSet> reconlib.Models.TrainingSets.TrainingSet [get], [set]`

The loaded training sets list

Definition at line 182 of file TrainingSets.cs.

The documentation for this class was generated from the following file:

- D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Models/ **TrainingSets.cs**

Chapter 7

File Documentation

7.1 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Analysis/Hocr/HocrAnalyzer.cs File Reference

Classes

- class **reconlib.Analysis.Hocr.HocrAnalyzer**
  Analysis (p. *9) and extraction of data from an hocr data structure*

Namespaces

- namespace **reconlib.Analysis.Hocr**

7.2 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Analysis/Hocr/HocrEntity.cs File Reference

Classes

- class **reconlib.Analysis.Hocr.HocrEntity**
  Represents an hocr data structure

Namespaces

- namespace **reconlib.Analysis.Hocr**

7.3 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Analysis/Hocr/Sections/Components/Baseline.cs File Reference

Classes

- class **reconlib.Analysis.Hocr.Sections.Components.Baseline**
  Represents an hocr baseline.

Namespaces

- namespace **reconlib.Analysis.Hocr.Sections.Components**

7.4 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Analysis/Hocr/Sections/Components/Bbox.cs File Reference

Classes

- class **reconlib.Analysis.Hocr.Sections.Components.Bbox**
  Represents an hocr bounding box

Namespaces

- namespace **reconlib.Analysis.Hocr.Sections.Components**

7.5 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Analysis/Hocr/Sections/OcrCarea.cs File Reference

Classes

- class **reconlib.Analysis.Hocr.Sections.OcrCarea**
  Represents an hocr content area

Namespaces

- namespace **reconlib.Analysis.Hocr.Sections**

7.6 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Analysis/Hocr/Sections/OcrLine.cs File Reference

Classes

- class **reconlib.Analysis.Hocr.Sections.OcrLine**
  Represents an hocr line

Namespaces

- namespace **reconlib.Analysis.Hocr.Sections**

7.7 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Analysis/Hocr/Sections/OcrPage.cs File Reference

Classes

- class **reconlib.Analysis.Hocr.Sections.OcrPage**
  Represents an hocr page structure

Namespaces

- namespace **reconlib.Analysis.Hocr.Sections**

7.8 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Analysis/Hocr/Sections/OcrPar.cs File Reference

Classes

- class **reconlib.Analysis.Hocr.Sections.OcrPar**
  Represents an hocr paragraph

Namespaces

- namespace **reconlib.Analysis.Hocr.Sections**

7.9 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Analysis/Hocr/Sections/OcrxWord.cs File Reference

Classes

- class **reconlib.Analysis.Hocr.Sections.OcrxWord**
  Represents an hocr word

Namespaces

- namespace **reconlib.Analysis.Hocr.Sections**

7.10 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Analysis/Levenshtein.cs File Reference

Classes

- class **reconlib.Analysis.Levenshtein**
  Levenshtein distance. Computes the distance between two words.

Namespaces

- namespace **reconlib.Analysis**

7.11 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Analysis/Mrz.cs File Reference

Classes

- class **reconlib.Analysis.Mrz**

  Represents an MRZ (machine readeable zone)

Namespaces

- namespace **reconlib.Analysis**

7.12 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Classification/OcKnnClassifier.cs File Reference

Classes

- class **reconlib.Classification.OcKnnClassifier**

  OcKnn classifier wrapper

Namespaces

- namespace **reconlib.Classification**

7.13 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Exceptions/EmptyFolderException.cs File Reference

Classes

- class **reconlib.Exceptions.EmptyFolderException**

  Exception meant to be thrown on empty folder

Namespaces

- namespace **reconlib.Exceptions**

7.14 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/↩ FeatureExtraction/Algorithms/GlobalFeaturesExtractionAlgorithm.cs File Reference

Classes

- class **reconlib.FeatureExtraction.Algorithms.GlobalFeaturesExtractionAlgorithm**

  Global features extraction wrapper Extracts the global features of an image

Namespaces

- namespace **reconlib.FeatureExtraction.Algorithms**

7.15 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/↩ FeatureExtraction/IFeatureExtractionAlgorithm.cs File Reference

Classes

- interface **reconlib.FeatureExtraction.IFeatureExtractionAlgorithm**

  Feature extraction algorithms interface

Namespaces

- namespace **reconlib.FeatureExtraction**

7.16 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/↩ Models/ClassificationEntity.cs File Reference

Classes

- class **reconlib.Models.ClassificationEntity**

  ClassificationEntity *(p. 22) represents the result data of a classification. This entity saves the*

  See also

  TrainingSet *(p. 65)*

  and the score obtained after the classification step

Namespaces

- namespace **reconlib.Models**

7.17 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Models/Config.cs File Reference

Classes

- class **reconlib.Models.Configuration**

  Global access configuration data
- class **reconlib.Models.Set**

  Represents an xml config set node
- class **reconlib.Models.Config**

  Represents the xml config root node

Namespaces

- namespace **reconlib.Models**

7.18 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Models/ExtractedData.cs File Reference

Classes

- class **reconlib.Models.ExtractedData**

  ExtractedData *(p. 26) represents the data extracted after the semantic analysis step It allows to save the extracted content and its confidence*

Namespaces

- namespace **reconlib.Models**

7.19 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Models/Patterns/AnalysisPattern.cs File Reference

Classes

- class **reconlib.Models.Patterns.AnalysisPattern**

  Represents an analysis pattern. Each document must have it's own pattern
- class **reconlib.Models.Patterns.Field**

  Represents a pattern field xml node
- class **reconlib.Models.Patterns.Pattern**

  Represents the pattern xml root node

Namespaces

- namespace **reconlib.Models.Patterns**

7.20 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/↩ Models/PreProcessedImage.cs File Reference

Classes

- class **reconlib.Models.PreProcessedImage**

  Represents a processed image (pre-processing filters applied)

Namespaces

- namespace **reconlib.Models**

7.21 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/↩ Models/ReconReport.cs File Reference

Classes

- class **reconlib.Models.ReconReport**

  Represents the report containing all the information from the system's execution

Namespaces

- namespace **reconlib.Models**

7.22 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/↩ Models/TrainingSets.cs File Reference

Classes

- class **reconlib.Models.Training**

  Global access configuration data
- class **reconlib.Models.TrainingSet**

  Represents a single training set xml node
- class **reconlib.Models.TrainingSets**

  Represents the training sets xml root node

Namespaces

- namespace **reconlib.Models**

7.23 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/obj/Debug/TemporaryGeneratedFile_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs File Reference

7.24 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/obj/Release/TemporaryGeneratedFile_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs File Reference

7.25 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/obj/Debug/TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs File Reference

7.26 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/obj/Release/TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs File Reference

7.27 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/obj/Debug/TemporaryGeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs File Reference

7.28 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/obj/Release/TemporaryGeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs File Reference

7.29 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/Ocr/Engines/Tesseract.cs File Reference

Classes

- class **reconlib.Ocr.Engines.Tesseract**

  Tesseract *(p. 63) engine wrapper*

Namespaces

- namespace **reconlib.Ocr.Engines**

7.30 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/↩ Ocr/IOcrAnalyzer.cs File Reference

Classes

- interface **reconlib.Ocr.IOcrAnalyzer**
  Ocr *(p. 12) wrappers interface*

Namespaces

- namespace **reconlib.Ocr**

7.31 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/↩ PreProcessing/Algorithms/AdaptiveSmooth.cs File Reference

Classes

- class **reconlib.PreProcessing.Algorithms.AdaptiveSmooth**
  Adaptative smooth filter

Namespaces

- namespace **reconlib.PreProcessing.Algorithms**

7.32 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/↩ PreProcessing/Algorithms/Binarization.cs File Reference

Classes

- class **reconlib.PreProcessing.Algorithms.Binarization**
  Binarization *(p. 20) filter.*

Namespaces

- namespace **reconlib.PreProcessing.Algorithms**

7.33 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/↩ PreProcessing/Algorithms/MedianFilter.cs File Reference

Classes

- class **reconlib.PreProcessing.Algorithms.MedianFilter**
  Median filter

Namespaces

- namespace **reconlib.PreProcessing.Algorithms**

7.34 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/PreProcessing/Algorithms/SkewCorrector.cs File Reference

Classes

- class **reconlib.PreProcessing.Algorithms.SkewCorrector**
  Image skew correction algorithm

Namespaces

- namespace **reconlib.PreProcessing.Algorithms**

7.35 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/PreProcessing/IPreProcessingAlgorithm.cs File Reference

Classes

- interface **reconlib.PreProcessing.IPreProcessingAlgorithm**
  Image pre-processing algorithms interface

Namespaces

- namespace **reconlib.PreProcessing**

7.36 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/ProcessDocument.cs File Reference

Classes

- class **reconlib.ProcessDocument**
  System's entry point.

Namespaces

- namespace **reconlib**

7.37 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/↩ Properties/AssemblyInfo.cs File Reference

7.38 D:/git_workspace/DI5_2016_2017_PRD/projet_recherche_developpement/reconlib/reconlib/↩ Tools/ImageTools.cs File Reference

Classes

- class **reconlib.Tools.ImageTools**

  Image handling tools

Namespaces

- namespace **reconlib.Tools**

Index

Comptes rendus hebdomadaires

Compte rendu n°1 du Mercredi 21 Septembre 2016

En attente de la réunion avec le correspondant de FoxNot, M. Alexis SEPCHAT, j'ai consulté quelques thèses à la BU [1] sur la reconnaissance de l'écriture manuscrite. Ne sachant pas exactement sur quoi porterai le projet (reconnaissance de mots manuscrits ou écriture informatisée), j'ai consulté les quelques publications de la BU sur le domaine. Les deux plus pertinentes que j'ai trouvé sont :

- Étude et réalisation d'un système adaptatif pour la reconnaissance en ligne de mots manuscrits
  - DUNEAU Laurent
  - DI_TH_383
- Contribution à la segmentation et à la Reconnaissance de l'Ecriture Manuscrite
  - DARGENTON Patrice
  - DI_TH_397

Je n'ai pas lu ces publications entièrement, seulement les grandes lignes afin de me familiariser avec le sujet. La réunion avec M. Sepchat et M. Slimane m'a permis de comprendre les objectifs du projet et de trouver réponse à quelques questions. Voici les notes que j'ai pu en tirer :

- partenariat Groupe Monassier (notaires)
- faciliter les transaction
  - faciliter la constitution du dossier
- document principal : titre de propriété
  - toutes les informations
  - Projet : lecture auto des documents afin de récupérer les données du particulier et aider la saisie
- attentes :
  - prendre quelques documents et être capable :
    - reconnaitre le document
    - lecture automatisée des informations
  - les documents ne sont pas "pareil" donc extractions sémantiques
    - détection de mots-clé et extractions
  - état de l'art
    - lecture automatique de documents
    - renseignement des solutions actuelles

- utilisation de briques de base + modules à ajouter afin de faire correspondre le produit au besoin
- pas d'existant
- équipements
  - idéal : scan du document avec appareil mobile
  - idée : déport du traitement coté serveur puis, plus tard, porter le traitement sur les machines client
  - technos
    - rabbitmq (traitement async)
    - worker pour traitement des documents
      - doit être async
- www-pp.foxnot.com - version beta

Suite à cette réunion, j'ai changé mes axes de recherche. Dans l'après-midi je me suis ainsi concentré sur les technologies d'OCR [2] existantes et potentiellement utilisables dans le projet. Lors de mes recherches, j'ai pu trouver un tableau comparatif [3] de certaines librairies OCR que m'a guidé dans une certaine mesure. Après étude du tableau et de quelques autres recherches, voici la liste des librairies actuelles potentiellement utilisables :

- Asprise OCR
- LeadTools
- Yunmai
- Tesseract
  - https://github.com/tesseract-ocr/tesseract
  - http://www.pixel-technology.com/freeware/tessnet2/

J'ai également effectué des recherches sur des publications plus adaptées aux attentes. Voici les deux les plus pertinentes trouvées :

- Improving classification using a Confidence Matrix based on weak classifiers applied to OCR [4]
  - Juan Ramon Rico-Juan, Jorge Calvo-Zaragoza
- Adaptative quality control of digital documents in mass digitization projects [5]
  - Ben Salah, Ahmed

Compte rendu n°2 du Jeudi 22 Septembre 2016

Durant la matinée j'ai continué la lecture des publications trouvées la veille. Suite à la réunion avec M. Ramel et M. Slimane, j'ai à nouveau modifié mes axes de recherche prenant compte les conseils reçus :

1. Bien identifier les problèmes en catégories
2. Rechercher des solutions pour chaque catégorie

Voici donc les catégories identifiées :

- Reconnaissance du document
- Repérage et analyse des ROI
  - Documents ayant des structures différentes mais contenant les mêmes informations
- OCR sur ces zones afin d'extraire les informations pertinentes

L'après-midi à été consacrée à des nouvelles recherches. J'ai ainsi pu trouver une publication qui pourrait apporter une solution à la catégorie "Reconnaissance du document" [6].

- Reconnaissance et extraction de documents : Une application industrielle à la détection de documents semi-structurés [7]
  - Olivier Augereau

J'ai également repéré certains algorithmes/méthodes sur lesquels je dois encore effectuer des recherches, notamment :

— SURF
— FLANN
— RANSAC

Quelques définitions trouvées :

— LAD : lecture automatique de documents
— RAD : reconnaissance automatique de documents
— GED : gestion électronique des documents

Compte rendu n°3 du Mercredi 28 Septembre 2016

La matinée à été consacrée à la lecture d'une partie de la thèse de The Anh Pham, *Détection robuste de jonctions et points d'intérêt dans les images et indexation rapide de caractéristiques dans un espace de grande dimension* [8]. Après étude, le contenu de cette thèse ne permet d'apporter aucune solution à la problématique posée par notre sujet. Durant l'après-midi, j'ai eu l'occasion de commencer la lecture de la thèse de Olivier AUGERAU [2], *Reconnaissance et classification d'images de documents* [9]. Cette thèse est particulièrement intéressante car le sujet étudié permet de répondre et d'apporter des solutions à une des problématiques de notre sujet (comment détecter si une image contient un document et de quel document il s'agit).

Compte rendu n°4 du Jeudi 29 Septembre 2016

Durant la matinée, j'ai pu continuer la lecture de la thèse de M. Augereau [2]. Cela à permis de confirmer les propos mentionnés précédemment. L'après-midi à été consacrée à une séance prévue avec M.Ragot et M.Soukhal sur les documents à apporter durant le projet, ainsi qu'une présentation sur les méthodes agiles.

Compte rendu n°5 du Mercredi 05 Octobre 2016

Pendant la matinée, j'ai continué la lecture de la thèse de M.Augereau [2]. L'après-midi à été consacrée au rapport. Ayant eu des soucis avec la classe *Polytech* de M.Aupetit, ainsi qu'avec le logiciel *MikTex* (plus tard dans la journée j'ai remarqué qu'il était devenu corrompu suite à la dernière mise à jour *Windows*), l'après-midi à été passée à mettre en place le système de compilation du rapport et la correction des soucis rencontrés.

Compte rendu n°6 du Jeudi 06 Octobre 2016

Durant cette journée j'ai eu l'occasion de basculer les rapports hebdomadaires sur le rapport final, et faire la mise en place du cahier des spécifications. Comme la base LaTeX de M.Ragot contient plusieurs soucis (notamment au niveau de l'encodage des caractères), la rédaction du cahier des spécifications sera effectuée sur *Word*. J'ai également eu l'occasion de continuer la lecture de la thèse de M.Augereau [2].

Compte rendu n°7 du Mercredi 12 Octobre 2016

La journée à été consacrée à la rédaction du rapport. Certaines parties ont été débutées (structurellement) et d'autres rédigées en intégralité (Introduction). J'ai également eu l'occasion d'avancer la lecture de la thèse de M.Augereau [2]. Durant l'après-midi, une réunion à été effectuée avec M. Ramel, au cours de laquelle certains points ont été éclaircis (technologies).

Compte rendu n°8 du Jeudi 13 Octobre 2016

Durant la matinée j'ai eu l'occasion d'avancer la rédaction du cahier des spécifications. J'ai également crée le *repository* qui contiendra le projet et basculé tous les documents. De plus, j'ai procédé à l'installation et prise en main des logiciels nécessaires par la suite du projet (PlantUml, Sonar, MSProject et MindView).

Compte rendu n°9 du Mercredi 19 Octobre 2016

Cette journée à été consacrée à la rédaction du cahier des spécifications. J'ai ainsi effectué quelques corrections et crée les diagrammes d'activité.

Compte rendu n°10 du Jeudi 20 Octobre 2016

Durant cette journée j'ai pu lire la publication [12]. De plus, j'ai également continué la correction du cahier des spécifications et commencer la préparation du planning prévisionnel.

Compte rendu n°11 du Mercredi 26 Octobre 2016

Durant cette journée j'ai eu l'occasion de continuer la mise en place du planning provisionnel et effectuer des recherches sur *RabbitMQ* et sa façon de fonctionner. Suite à la réunion avec M.Sepchat, j'ai commencé la mise à jour du CDS avec les nouvelles informations.

Compte rendu n°12 du Mercredi 09 Novembre 2016

La journée du 09 Novembre à été consacrée à la restructuration et correction des erreurs signalées sur le CDS lors de la réunion avec M.Soukhal, responsable des cahier de spécifications, ainsi qu'à la rédaction des parties manquantes.

Compte rendu n°13 du Mercredi 16 Novembre 2016

Lors de cette journée, j'ai pu étudier [13]. Suite à la réunion avec M. Sepchat et M.Slimane, j'ai pu commencer la veille technologique sur Mono : compatibilités, modes de fonctionnement. J'ai également continué la prise de notes pour la rédaction du rapport.

Compte rendu n°14 du Jeudi 17 Novembre 2016

La journée du 17 à été consacrée à l'installation de la VM Debian et les essais sur Mono. Suite à mes recherches, j'ai pu déterminer un mode d'installation, défini ci-dessous. Mono s'installe assez facilement sur Linux (Debian 8 jessie) malgré quelques étapes plus difficiles à comprendre. Si on suit la procédure indiquée sur le site de Mono, on se rends compte que certaines dépendances n'y sont pas mentionnées et doivent quand-même être installées pour le bon fonctionnement de Mono. Après quelques recherches, j'ai pu surmonter ces problèmes et définir une procédure d'installation (détaillée en annexe). Une fois Mono installé, j'ai installé MonoDevelop, un IDE permettant de manipuler Mono et accélérer le processus de développement. Ici également quelques soucis d'installation, des dépendances qui ne sont pas mentionnées. Quelques autres recherches m'ont également permis de surmonter ces problèmes. Au final, l'IDE nécessite quelques réglages (notamment au niveau de la sortie console) mais compile et lance des programmes simples sans problèmes jusqu'à présent.

Compte rendu n°15 du Mercredi 30 Novembre 2016

Cette journée à été consacrée à la correction du CDS. Suite aux remarques de M.Sepchat et M.Ramel, ainsi que à la réunion avec M.Ramel durant l'après-midi, j'ai pu effectuer les corrections textuelles nécessaires et commencer la correction des diagrammes. Suite à lé réunion avec M.Billaut et M.Martineau concernant les plannings, j'ai également commencé la correction du planning prévisionnel.

Compte rendu n°16 du Jeudi 01 Décembre 2016

Le jeudi à été consacré à la finalisation du CDS et du plan de développement. J'ai également effectué des recherches afin de compléter mes notes concernant l'état de l'art.

Compte rendu n°17 du Mercredi 07 Décembre 2016

Rédaction du rapport.

Compte rendu n°18 du Jeudi 08 Décembre 2016

Finalisation du rapport et du poster.

Compte rendu n°19 du Mercredi 04 Janvier 2017

Début de la phase d'analyse et réflexion sur les diagrammes UML

Compte rendu n°20 du Jeudi 05 Janvier 2017

Progression dans l'analyse et rédaction des diagrammes UML

Compte rendu n°21 du Mercredi 11 Janvier 2017

Progression dans l'analyse et rédaction des diagrammes UML. Étude et tests des frameworks Aforge/Accord sous Mono.

Compte rendu n°22 du Jeudi 12 Janvier 2017

Refonte de la VM Debian

Compte rendu n°23 du Mercredi 18 Janvier 2017

Préparation du projet et étude des dépendances nécessaires. Correction de la VM Debian à nouveau.

Compte rendu n°24 du Jeudi 19 Janvier 2017

Pour cause de défaillances de la VM Debian, refonte du projet sous Windows. Étude et analyse du code fourni par l'équipe RFAI.

Compte rendu n°25 du Mercredi 25 Janvier 2017

Préparation de la base d'apprentissage et début du développement de la fonctionnalité de pré-traitement.

Compte rendu n°26 du Jeudi 26 Janvier 2017

Progression dans le développement de la fonctionnalité et début du développement de la fonctionnalité de chargement des images.

Compte rendu n°27 du Mercredi 01 Février 2017

Finalisation de la fonctionnalité de chargement et pré-traitement des images. Étude des bibliothèques *GlobalFeatures* et *Saillant Primitives* et début du développement de la fonctionnalité d'extraction des caractéristiques.

Compte rendu n°28 du Jeudi 02 Février 2017

Progression dans le développement de la fonctionnalité d'extraction des caractéristiques et mise à jour des diagrammes UML

Compte rendu n°29 du Mercredi 08 Février 2017

Finalisation de la fonctionnalité d'extraction des caractéristiques.

Compte rendu n°30 du Jeudi 09 Février 2017

Étude de la bibliothèque *OcKnn* de l'équipe RFAI et début du développement de la fonctionnalité de classification.

Compte rendu n°31 du Mercredi 15 Février 2017

Étude du code fourni par l'équipe RFAI et étude du moteur OCR Tesseract.

Compte rendu n°32 du Jeudi 16 Février 2017

Début du développement de la fonctionnalité OCR et correction de la fonctionnalité de prétraitement. Mise à jour des diagrammes UML.

Compte rendu n°33 du Mercredi 01 Mars 2017

Progression dans le développement de la fonctionnalité OCR

Compte rendu n°34 du Jeudi 02 Mars 2017

Étude du code fourni par l'équipe RFAI et déboggage de la bibliothèque de classification :

- Soucis rencontrés :
  - Erreur lors de l'intégration de la bibliothèque *IntrinsicClassification* → Recompilation
  - *Learning Set* vidé lors de la création de l'objet

Début du développement de la fonctionnalité d'analyse HOCR.

Compte rendu n°35 du Mercredi 08 Mars 2017

Progression dans le développement de la fonctionnalité d'analyse hocr et mise en place de la structure de regroupement des données hocr.

Compte rendu n°36 du Jeudi 09 Mars 2017

Début de la campagne de tests et intégration de Sonar à visual studio pour un retour des métriques en temps réel.

Compte rendu n°37 du Mercredi 15 Mars 2017

Travail effectué sur la fonctionnalité d'analyse et extraction sémantique.

Compte rendu n°38 du Jeudi 16 Mars 2017

Finalisation de la fonctionnalité de classification.

Compte rendu n°39 du Mercredi 22 Mars 2017

Finalisation de la fonctionnalité d'analyse et extraction sémantique

Compte rendu n°40 du Jeudi 23 Mars 2017

Progression sur les tests unitaires et d'intégration, mise à jour des diagrammes et finalisation de la documentation.

Compte rendu n°41 du Mercredi 29 Mars 2017
Rédaction du rapport

Compte rendu n°42 du Jeudi 30 Mars 2017
Rédaction du rapport

Webographie

[WWW1] Abbyy. *Adaptive Document Recognition Technology (ADRT)*. 2016. URL : https://abbyy.technology/en:features:ocr:adrt (visité le 21/09/2016).

[WWW2] Abbyy. *FineReader*. 2016. URL : https://www.abbyy.com/fr-fr/finereader/ (visité le 21/09/2016).

[WWW3] Olivier Augereau. *RLSA*. 2011. URL : http://www.olivier-augereau.com/blog/?p=15 (visité le 24/11/2016).

[WWW4] Olivier Augereau. *SIFT et SURF*. 2011. URL : http://www.olivier-augereau.com/blog/?p=73 (visité le 26/11/2016).

[WWW5] charlesw. *A .Net wrapper for tesseract-ocr*. 2016. URL : https://github.com/charlesw/tesseract (visité le 21/09/2016).

[WWW6] Chebert. *Introduction à RabbitMQ - AMQP Partie I*. 2010. URL : https://blog.zenika.com/2010/12/19/introduction-a-rabbitmq-amqp/ (visité le 26/10/2016)

[WWW7] Chebert. *Introduction à RabbitMQ - RabbitMQ, le vif du sujet*. 2011. URL : https://blog.zenika.com/2011/04/26/introduction-a-rabbitmq-le-vif-du-sujet/ (visité le 26/10/2016).

[WWW8] Erlang. *Build massively scalable soft real-time systems*. 2016. URL : https://www.erlang.org (visité le 26/10/2016).

[WWW9] FLANN. *Fast Library for Approximate Nearest Neighbors*. 2016. URL : http://www.cs.ubc.ca/research/flann/ (visité le 21/09/2016).

[WWW10] Accord Framework. *DocumentSkewChecker Class*. 2017. URL : http://accord-framework.net/docs/html/T_Accord_Imaging_DocumentSkewChecker.htm (visité le 28/03/2017).

[WWW11] Accord Framework. *Median Class*. 2017. URL : http://accord-framework.net/docs/html/T_Accord_Imaging_Filters_Median.htm (visité le 28/03/2017).

[WWW12] Accord Framework. *SISThreshold Class*. 2017. URL : http://accord-framework.net/docs/html/T_Accord_Imaging_Filters_SISThreshold.htm (visité le 28/03/2017).

[WWW13] Accord.NET Framework. *Machine learning made in a minute*. 2016. URL : http://accord-framework.net/ (visité le 12/10/2016).

[WWW14] Accord.NET Framework. *Getting started*. 2017. URL : https://github.com/accord-net/framework/wiki/Getting%20started (visité le 12/10/2016).

[WWW15] Aforge.NET Framework. *CloudsTexture Class*. 2006-2013. URL : http://www.aforgenet.com/framework/docs/html/587ea20e-c329-6ae8-98a3-e495aa4fc18e.htm (visité le 12/01/2017).

[WWW16] Aforge.NET Framework. *Convolution Class*. 2006-2013. URL : http://www.aforgenet.com/framework/docs/html/3b7d8422-2d79-8019-8933-a0472040c124.htm (visité le 12/01/2017).

[WWW17] Aforge.NET Framework. *Aforge.NET Framework*. 2016. URL : http://www.aforgenet.com/ (visité le 12/10/2016).

[WWW18] Md. Abul Hasnat. *Run Length Smoothing Algorithm (RLSA)*. 2007. URL : http://crblpocr.blogspot.fr/2007/06/run-length-smoothing-algorithm-rlsa.html (visité le 24/11/2016).

[WWW19] Docker Inc. *Build, Ship, Run*. 2016. URL : https://www.docker.com/ (visité le 09/11/2016).

[WWW20] Docker Inc. *Build, Ship, Run*. 2016. URL : https://www.docker.com/what-docker (visité le 09/11/2016).

[WWW21] Lovisa Johansson. *Part1 : RabbitMQ for beginners - What is RabbitMQ*. 2015. URL : https://www.cloudamqp.com/blog/2015-05-18-part1-rabbitmq-for-beginners-what-is-rabbitmq.html (visité le 26/10/2016).

[WWW22] Microsoft. *Commentaires de documentation XML*. 2017. URL : https://msdn.microsoft.com/fr-fr/library/b2s063f7.aspx (visité le 18/01/2017).

[WWW23] Microsoft. *Conventions de codage CSharp*. 2017. URL : https://msdn.microsoft.com/fr-fr/library/ff926074.aspx (visité le 18/01/2017).

[WWW24] Nuance. *Solutions OmniPage pour vous et votre entreprise*. 2016. URL : http://www.nuance.fr/for-individuals/by-product/omnipage/index.htm (visité le 06/11/2016).

[WWW25] Oasis. *AMQP - Advanced Message Queuing Protocol*. 2016. URL : http://www.amqp.org (visité le 26/10/2016).

[WWW26] Pivotal. *RabbitMQ*. 2016. URL : https://www.rabbitmq.com/ (visité le 26/10/2016)

[WWW27] Mono Project. *Cross platform, open source .NET framework*. 2016. URL : http://www.mono-project.com/ (visité le 17/11/2016).

[WWW28] ETH - Swiss Federal Institute of Technology Zurich. *SURF : speeded up robust features*. URL : http://www.vision.ee.ethz.ch/~surf/index.html (visité le 26/11/2016).

[WWW29] tesseract-ocr. *Improve Quality*. 2016. URL : https://github.com/tesseract-ocr/tesseract/wiki/ImproveQuality (visité le 21/09/2016).

[WWW30] tesseract-ocr. *Tesseract Open Source OCR Engine*. 2016. URL : https://github.com/tesseract-ocr/tesseract (visité le 21/09/2016).

[WWW31] tesseract-ocr. *Tesseract Open Source OCR Engine Documentation*. 2016. URL : https://github.com/tesseract-ocr/tesseract/wiki (visité le 21/09/2016).

[WWW32] tmbdev. *hocr-tools*. 2016. URL : https://github.com/tmbdev/hocr-tools (visité le 30/11/2016).

[WWW33] tmbdev. *ocropy*. 2016. URL : https://github.com/tmbdev/ocropy (visité le 30/11/2016).

[WWW34] Wikipédia. *Filtre médian*. 2016. url : https://fr.wikipedia.org/wiki/Filtre_m%C3%A9dian (visité le 30/11/2016).

Bibliographie

[1] Fahimeh Alaei, Nathalie Girard, Sabine Barrat et Jean-Yves Ramel. *A New One-class Classification Method Based on Symbolic Representation : Application to Document Classification*.

[2] Olivier Augereau. « Reconnaissance et classification d'images de documents ». Université Bordeaux 1, 2013.

[3] Herbert Bay, Andreas Ess, Tinne Tuytelaars et Luc Van Gool. *Speeded-Up Robust Features (SURF)*. 2008.

[4] Stefano Ferilli, Fabio Leuzzi, Fulvio Rotella et Floriana Esposito. *A run length smoothing-based algorithm for non-manhattan document segmentation*.

[5] Mohamadally Hasan et Fomani Boris. *SVM : Machines à Vecteurs de Support ou Séparateurs à Vastes Marges*. 2006.

[6] David G. Lowe. *Distinctive Image Features from Scale-Invariant Keypoints*. 2004.

[7] Robert C. Martin. *Coder Proprement*. 2009.

[8] Stephen O'Hara et Bruce A. Draper. *Introduction to the bag of features paradigm for image classification and retrieval*. 2011.

[9] P.Bonnet. *Filtrage Médian*.

[10] Daniel Rosner, Costin-Anton Boiangiu, Mihai Zaharescu et Ion Bucur. *Image Skew Detection : A Comprehensive Study*.

[11] J. Sauvola et M. PietikaKinen. *Adaptive document image binarization*. 1998.

[12] Nicolas Sidère, Jean-Yves Ramel, Sabine Barrat, Vincent Poulain D Andecy et Saddok Kebairi. *A Compliant Document Image Classification System based on One-Class Classifier*.

[13] Nicolas Sidère, Jean-Yves Ramel, Sabine Barrat, Vincent Poulain D Andecy et Saddok Kebairi. *Document identification using one-class classifier*.

Projet Recherche & Développement

Reconnaissance d'éléments textuels ou graphiques dans les documents juridiques numérisés

Horacio Cachinho

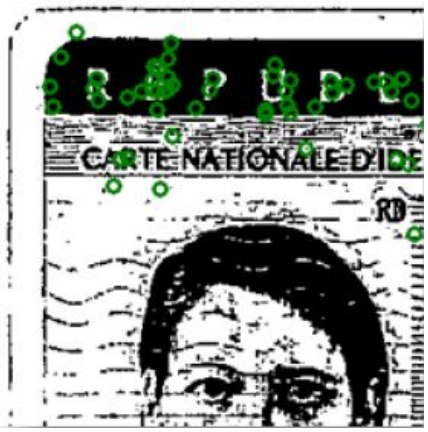
Encadrement : Jean-Yves Ramel et Mohamed Slimane

FOXNOT

En collaboration avecFoxNot

Objectifs

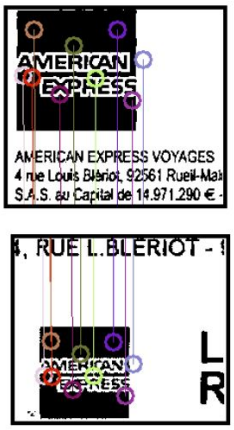
Créer un *framework* de **reconnaissance de documents juridiques** regroupant des services d'extraction et de reconnaissance automatique de données d'intérêt.



Détection des points d'intérêt

Utilité

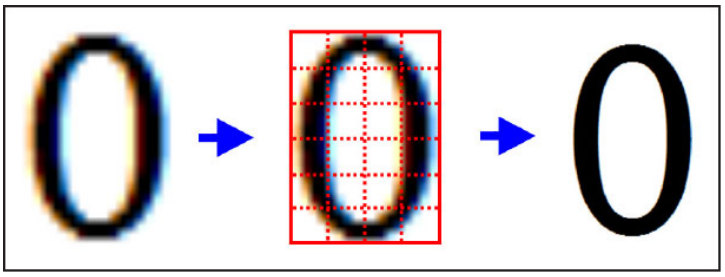
Permettre aux utilisateurs du service **FoxNot** de pouvoir **numériser** et/ou **capturer** des **documents administratifs** pour ensuite en **extraire des informations spécifiques automatiquement** (noms, adresse, dates).



Mise en correspondance des documents

Technologies

- Tesseract
- Accord.Net
- Aforge.Net
- RabbitMQ
- Docker
- Linux Debian



Optical Character Recognition - OCR

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

POLYTECH TOURS

Informatique

Projet Recherche & Développement

Reconnaissance d'éléments textuels ou graphiques dans les documents juridiques numérisés

Horacio Cachinho

Encadrement : Jean-Yves Ramel et Mohamed Slimane

FOXNOT

En collaboration avec FoxNot

Objectifs

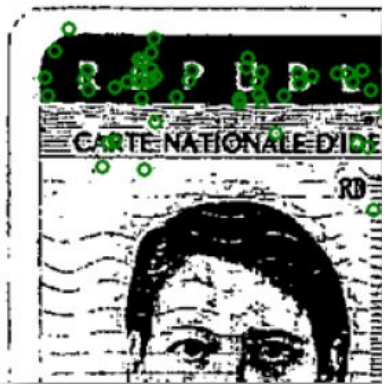
Créer un *framework* de **reconnaissance de documents juridiques** regroupant des services d'extraction et de reconnaissance automatique de données d'intérêt.

Utilité

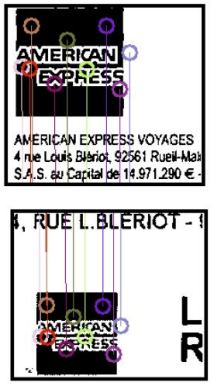
Permettre aux utilisateurs du service **FoxNot** de pouvoir **numériser** et/ou **capturer** des **documents administratifs** pour ensuite en **extraire des informations spécifiques automatiquement** (noms, adresse, dates).

Technologies

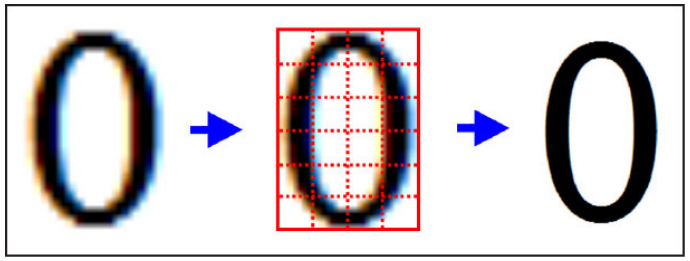
- Tesseract
- Accord.Net
- Aforge.Net
- RabbitMQ
- Docker
- Linux Debian



Détection des points d'intérêt



Mise en correspondance des documents



Optical Character Recognition - OCR

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

POLYTECH TOURS

Informatique

Reconnaissance d'éléments textuels ou graphiques dans les documents juridiques numérisés

Résumé

Lors de la rédaction d'actes notariés, un notaire peut nécessiter d'un grand nombre d'informations et de documents concernant les personnes et/ou biens concernés. La récupération, échange et traitement de ces documents peut ainsi s'avérer longue et fastidieuse (document dupliqué ou manquant, illisible ou erroné) tant du côté du cabinet de notaires que de celui des particuliers. De plus, certains documents possèdent en partie les mêmes informations (identification, adresse, etc.), ce qui crée une grande redondance des données. Ce projet vise ainsi à débuter le développement d'un *framework* regroupant des services permettant la reconnaissance et l'extraction de données d'intérêt (sous forme textuelle ou graphique) sur des documents juridiques préalablement dématérialisés (scan ou photo) afin de faciliter l'échange d'informations entre particuliers et notaires et la constitution du dossier.

Mots-clés

Reconnaissance documents, Numérisation, Documents juridiques, OCR

Abstract

When establishing notarial deeds, the solicitor may require lots of informations and several documents related to the concerned persons and / or goods. Thus, the retrieval, exchange and processing of these documents can be long and tedious (duplicated or missing document). In addition, some documents have the same information (identification, address, etc.), which creates great data redundancy. This project aims to development a *framework* grouping together services allowing the recognition and extraction of data of interest (in textual or graphical forms) on previously dematerialized legal documents (scan or photo), in order to facilitate the exchange of information.

Keywords

Document recognition, Scan, Legal documents, OCR

Entreprise
FoxNot

Tuteurs entreprise
Thierry Arnaly
Dominique Bazin

Étudiant
Horacio Cachinho (DI5)

Tuteurs académiques
Jean-Yves Ramel
Mohamed Slimane