Pictwriters

Soutenance 2
Projet OCR

Archer Stéphane, Eychenne Mélanie,
Lorenzini Ugo et Petrov Daniel

13 Décembre 2013

TABLE DES MATIÈRES

CHAPITRE 1

INTRODUCTION

Ce document rassemble toutes les informations sur l'élaboration de notre projet de SPE à EPITA. Celui ci est la conception d'un **O**ptical **C**haracter **R**ecognition (Reconnaissance Optique de Caractères en français) que l'on nomme : O.C.R. Ce projet est codé en Objective Caml (aussi nommé OCaml) et est développé sous FreeBSD.

Pictwritters, les écrivains de l'image :

Nous avons eu environ 3 mois (de Septembre à Décembre) pour réaliser ce projet. Il nous a fallut être ordonnés et vifs afin de le concevoir dans les meilleures conditions.

Mais concrètement, vous devez vous demander : ***"qu'est ce qu'un O.C.R ?"***

Un O.C.R est un logiciel permettant le traitement d'un texte présent dans une image telle que : une photographie, un scan d'un livre, etc. Après ce traitement, l'O.C.R permet d'obtenir un document avec le texte brut modifiable.

Principe d'utilisation de ce logiciel :

L'utilisateur souhaite récupérer un fichier contenant le texte brut d'une image ou d'une numérisation d'une page. Ainsi, lorsqu'il va scanner son document, notre O.C.R. va lui permettre de transformer celui ci en un texte brut grâce à la liste des symboles.

CHAPITRE 2

PRÉSENTATION DES MEMBRES DE LA PICTWRITERS TEAM

Nous avons choisi comme nom de groupe la Pictwriters Team car nous allons permettre aux utilisateurs de notre logiciel de transformer leurs images contenant du texte ("Pict = picture) en un document contenant ce texte pouvant être modifié.

2.1 Stéphane

Bonjour, je m'appelle Stéphane ARCHER et je suis en SPE à EPITA. L'informatique a toujours été ma passion. De plus, ce projet est l'occasion pour moi de vérifier mes compétences de gestion d'un projet car on m'a nommé chef de projet. En début d'année je ne connaissais ni Daniel ni Ugo mais j'ai trouvé deux coéquipiers très sympathiques et sérieux tout comme Mélanie. Chaque membre du

groupe a su apporter sa contribution dans la joie et la bonne humeur. J'attends essentiellement de ce projet de pouvoir perfectionner ma programmation, apprendre à structurer un projet, répartir les tâches et vérifier que tous s'imbrique. Ce travail d'équipe devrait permettre à chacun de développer ses talents. On est tous pressés de pouvoir utiliser notre O.C.R et de le partager.

2.2 Mélanie

Seule représentante de la gente féminine du groupe, j'espère apporter un petit plus du côté de l'interface graphique dont on s'occupera pour la dernière soutenance. Ce projet est, selon moi, très intéressant car il m'est arrivé de nombreuses fois de me retrouver devant un scan d'une feuille sur internet que j'aurais aimé ajouter dans mon document avec quelques modifications. Mais, à chaque fois, j'étais obligée de recopier ce dont j'avais besoin. Grâce à ce projet, j'aurais "mon propre" O.C.R que je pourrai utiliser quand bon me semble!

2.3 Ugo

Moi c'est Ugo, j'ai commencé l'informatique avec l'ordinateur familial et je me suis tout de suite intéressé aux jeux vidéo. Plus tard j'ai eu envie de comprendre comment les choses marchaient, en particulier dans un ordinateur, et ce comment étaient conçus les programmes et jeux que j'utilisais. J'ai fait un peu de C quand j'étais encore qu'un débutant en informatique mais j'ai vite arrêté car faire des jeux en mode console sous Windows n'était pas si facile que ça et pas non plus très passionnant. L'année dernière j'ai été très emballé par le projet de SUP et plutôt satisfait du résultat. Cette année, le projet d'OCR à l'air d'être un projet intéressant, en partie la partie sur le réseau de neurones car je me suis toujours demandé comment fonctionnait une "intelligence artificielle" même si je suis conscient que cela sera dur et que je devrai acquérir de nouvelles compétences pour y arriver.

2.4 Daniel

Je me nomme Daniel PETROV et je suis un étudiant à l'EPITA. On peut dire que dès ma plus petite enfance j'ai été baigné dans le monde des ordinateurs. A mes trois ans j'étais déjà planté devant un ordinateur et j'avais soif d'apprendre et de m'amuser grâce aux logiciels ADIBOU. Et d'ailleurs, c'était l'unique façon pour mes parents de calmer mes pleurs et mon mauvais tempérament (et surtout de donner des moments de répits aux voisins qui m'entendaient crier et pleurer à tue-tête et je peux vous dire que j'avais de la voix). J'ai toujours été intrigué par les ordinateurs, par leur fonctionnement, par leur utilisation et tout ce qu'on peut faire avec. Ils m'ouvraient de nouveaux horizons qui m'étaient encore inconnus. J'y passais des journées et des nuits entières et combien de fois je me suis fait gronder par mes parents car j'avais complètement mis en pièces mon ordinateur pour justement comprendre ses composantes. Peu de temps après, j'ai commencé à m'intéresser au multimédia et tout ce qui en découle ; le son, les images, le design, etc. Arrivé en seconde année à L'EPITA, j'ai vu que le projet de l'année était de faire un OCR (Optical Character Recognition) et j'en ai été véritablement ravi. Cela me donnerait la possibilité de travailler sur cette problématique et surtout de réfléchir et réaliser le traitement de l'image du projet OCR.

CHAPITRE 3

ORGANISATION SUR L'ENSEMBLE DU SEMESTRE

3.1 Répartition des tâches pour la première soutenance

Nous avions rapidement effectué une répartition des tâches afin de pouvoir commencer notre projet le plus tôt possible. Voici un tableau montrant quelle personne s'était le plus occupée de telle ou telle partie.

Soutenance 1	
Pré-traitement de l'image	Daniel
Passage en niveau de gris	Mélanie
Rotation de l'image	Stéphane
Détection des zones de textes	Ugo
Site web	Mélanie
Rapport de la soutenance	Mélanie

3.2 Répartition des tâches pour la seconde soutenance

Pour cette soutenance, nous nous sommes répartis les tâches après notre semaine de contrôles. Cette répartition des tâches s'est faite très simplement : chaque personne voulant finir ce qu'elle avait commencé nous avons du simplement décider qui s'occuperait du réseau de neurones et de l'interface graphique. Étant donné que se sont deux parties importantes nous avons décidé que les meilleurs codeurs du groupe devaient s'en occuper avec, bien entendue l'aide des autres membres.

Soutenance 2	
Amélioration du Pré-traitement de l'image	Daniel
Amélioration de la détection des zones de textes	Ugo
Amélioration de la détection de l'angle de rotation	Stéphane
Interface graphique	Stéphane
Participation à la conception de l'interface graphique	Mélanie
Réseau de neurones	Ugo
Site web	Mélanie
Participation à la conception du site web	Stéphane
Rapport de la soutenance	Mélanie

3.3 Organisation générale

3.3.1 Première soutenance

Après avoir fait la répartition des tâches nous avions fait un dépôt de notre code source sur GIT. Nous étions restés en contact permanent grâce à Facebook mais aussi Skype et nous nous étions donnés quelques rendez-vous à EPITA pour voir la progression du projet et apporter de l'aide aux personnes en ayant besoin.

Il faut l'avouer, il avait été difficile pour nous de nous voir car certaines personnes de l'équipe habitent loin d'EPITA. Ainsi, par un souci de rentabilité de temps, nous avions décidé de nous donner rendez-vous dans une salle à EPITA que dans la semaine à la fin des cours (le plus souvent le vendredi car c'est le jour où nous terminons le plus tôt). Mais de toutes façons, nous savions que nous pouvions avoir un contact permanent grâce à nos portables et aux réseaux sociaux comme dit précédemment.

3.3.2 Dernière soutenance

Notre organisation n'a pratiquement pas changé. Nous avons eu tout de même plus de difficultés pour nous voir afin de faire un bilan d'où nous en étions dans le développement du projet car nous nous préparions également pour nos partiels de la semaine qui suivait.

CHAPITRE 4

VALIDATION DU CAHIER DES CHARGES

4.1 Cadre ✔

Le projet a bien été réalisé par une équipe de quatre personne et dans les temps demandés.

4.2 Restrictions ✔

Notre projet a été développé sous FreeBSD et a été codé en Objective Caml comme demandé. Nous l'avons trouvé très enrichissant.

4.3 Protocole ✔

Nous avons choisi notre nom de projet et de groupe (Pictwritters) et nous avons désigné notre chef de projet (Stéphane) dans les temps impartis.

4.4 Les soutenances ✔

Notre première soutenance à duré un peu plus de 14 minutes. Nous avons bien réussi à être dans les temps grâce à de nombreuses répétitions. Nous avons également eu le temps, pour cette seconde soutenance de nous préparer pour notre oral : nous espérons qu'elle se déroulera aussi bien que la première.

4.5 Règles à respecter ✔

- Pour le code
- Indentation des fonctions.
- Les noms de fonctions, variables, constantes et macros sont en anglais.
- Conception d'une hiérarchie correcte et cohérente des modules ne contenant pas de cycle.
- Les modules ont été utilisés avec la notation pointée (Module.fonction) ; la directive open était prohibée.
- 80 caractères par ligne
- Pas d'espace en fin de ligne dans les fichiers .ml et .mli
- Pas de Warning à la compilation sur notre code.

- Pour le projet
- Nous avons utilisé ocamlbuild. Nous avons fait un Makefile pour simplifier les choses.
- Le projet contient un fichier README qui explique comment compiler et utiliser notre projet.

4.6 Découpage du projet

1. Pré-traitement de l'image de départ, numérisée à partir d'un document A4, par l'effacement de bruits afin de produire un document parfait. FAIT
2. Extraction des lignes et des caractères. FAIT
3. Codage des caractéristiques d'une image d'un caractère, et apprentissage machine des caractéristiques de chaque symbole à détecter. NON FAIT
4. Développement de l'interface visuelle de l'application, interprétation et affichage de la mise en forme du document détecté antérieurement et définition de balises de mise en forme du document texte (le format des balises en html). FAIT

4.7 Calendrier des soutenances ✔

Toutes les tâches demandées ont été réalisées dans les délais. Nous vous invitons à regarder plus haut, dans ce rapport, afin de voir quelles personnes se sont occupées de telle ou telle partie (chapitre 3).

4.8 Pièces à fournir ✔

4.8.1 Première soutenance

- Un plan de soutenance
- Un rapport de soutenance
- Un site Web (et même deux pour nous ! ! Un Wordpress et un autre fait en langage HTML et CSS)

4.8.2 Seconde soutenance

- Un plan de soutenance
- Un rapport de projet
- Un dossier d'exploitation (manuel d'utilisation et manuel d'installation)
- Le projet (un exécutable)
- Un site Web (et toujours deux pour nous : le Wordpress et celui en langage HTML et CSS)

4.9 Bilan

En résumé tout à été fait mis à par le réseau de neurones que nous n'avons, malheureusement, pas réussi à implémenter malgré de nombreuses heures de recherches...

CHAPITRE 5

PRÉ-TRAITEMENT DE L'IMAGE

5.1 Première soutenance

5.1.1 Rappels

Voici des termes important dont nous allons vous parler dans cette partie du rapport :

Une image : On peut comparer une image à une matrice composée de pixels. En effet, pour modifier une image, il suffit de parcourir chaque pixel de l'image grâce à deux boucles for permettant d'accéder à la position (w,h) avec w : la largeur et h : la hauteur de l'image (comme pour accéder à une valeur d'une matrice).

Un pixel : La couleur d'un pixel peut être modifié grâce à ses composantes : r, g et b (r représentant le rouge, g représentant le vert et b représentant le bleu). On peut également modifier l'opacité de l'image grâce à rgba (lorsque a = 1 l'image est opaque).

Le bruit d'une image : Le bruit représente tout ce dont on ne veut pas garder dans l'image. Nous, nous souhaitons récupérer les caractères présents dans une image (et seulement les caractères).

5.1.2 But de l'étape

Pour faire un O.C.R il faut, tout d'abord, commencer par transformer l'image d'origine pour que la reconnaissance des caractères soit plus simple. En effet, notre but est de faire ressortir le tracé des caractères.

5.1.3 Principe

Pour faire ressortir le tracé des caractères, il faut éliminer (et même plus : éradiquer !!) le bruit de l'image et amplifier les marques des caractères. Pour cela, il existe plusieurs méthodes d'implémentation.

Chargement de l'image

Avant de pouvoir traiter notre image, il faut qu'on la charge. Pour cela nous avons utilisé la bibliothèque OCamlSDL car elle permet facilement de convertir une image en une matrice de pixels grâce à cette fonction :

let image = Sdlloader.load_image filename;;

Passage en niveau de gris

Tout d'abord, nous avons passé notre image en niveau de gris. La technique est simple : il suffit d'appliquer à chaque pixel de l'image la formule de luminosité moyenne suivante : 0.3 * rouge + 0.59 * vert + 0.11 * bleu.

Il existe également une autre technique qui est de faire la somme de chaque composante d'un pixel (c'est à dire r, g et b dont les valeurs sont comprises entre 0 et 255) et de la diviser par trois (ce qui donne un nouveau nombre compris entre 0 et 255). Mais elle ne prend pas en compte que notre œil n'a pas la même sensibilité pour chaque couleur. C'est pour cela qu'il vaut mieux éviter de mettre les mêmes valeurs pour les trois composantes d'un pixel d'une image pour la passer en niveau de gris.

$$Gris = \frac{(Rouge + Vert + Bleu)}{3}$$

Élimination du bruit de l'image

Puis, nous avons détecté et éliminé le bruit de l'image c'est à dire que nous avons enlevé les éléments inutiles de l'image pour faire ressortir les caractères. Pour cela, nous avons implémenté une fonction qui lorsqu'elle parcourt chaque pixel d'une image, si ce pixel est noir et entouré de pixels blancs alors il devient blanc. Plus concrètement, la fonction parcourt l'image : si le pixel est noir (à une valeur de 0) elle vérifie si les pixels qui l'entourent sont blancs (ont une valeur de 255). Si c'est le cas, elle modifie le pixel noir et le met en blanc, sinon elle passe au pixel suivant.

Binarisation

Et enfin, nous avons passé l'image en noir et blanc. Il existe différentes façons pour implémenter cette fonction. Nous avons décidé de faire la plus efficace mais également la plus simple. Il suffit de partir de l'image passée en niveau de gris et avec l'élimination du bruit et en fonction d'un certain seuil de luminosité on choisit de passer le pixel en noir "pur" ou en blanc "pur". Nous avons choisit comme valeur de seuil : 220.

Ainsi, pour chaque pixel rencontré lors du parcours de l'image si le pixel courant et supérieur à ce seuil on le met en blanc sinon on le passe en noir.

5.1.4 Bilan des étapes du pré-traitement de l'image

Voici l'image sans modification :

Tout d'abord, on fait le passage en niveau de gris de l'image :

Puis on élimine le bruit de l'image et on la passe en noir et blanc :

5.2 Changements pour la seconde soutenance

Pour cette soutenance nous avons implémenté un nouveau filtre.

Celui-ci est un filtre passe haut qui permet de mieux faire ressortir les contours de chaque symbole. On s'est inspiré d'un modèle de "matrice de convolution" pour le faire. Nous sommes assez satisfait du résultat car il est bon : il permet de bien découper les caractères entre eux pour faciliter la segmentation.

On a du, par ailleurs, inverser la binarisation car le filtre nous faisait ressortir les caractères en blanc, mais cela permet aussi la suppression du bruit isolé quand on le combine au filtre médian précédemment utilisé.

CHAPITRE 6

ROTATION DE L'IMAGE

6.1 Son but

Quand on scanne une page, celle ci n'est pas toujours "droite" ou l'écriture n'est pas forcément mis à l'horizontale.

Ainsi la reconnaissance des caractères est difficile.

C'est pour cela qu'en pré-traitement nous avons implémenté une fonction permettant une rotation de l'image.

Pour cela, il faut détecter l'angle d'inclinaison des lignes de texte puis se servir des informations récupérées afin de faire la rotation de l'image.

6.2 Son principe

Pour faire une rotation d'image il faut tout d'abord trouver l'angle d'inclinaison du texte et utiliser cet angle de rotation pour mettre le texte "droit".

6.2.1 Trouver l'angle d'inclinaison

La partie la plus difficile de la rotation est sans conteste de trouver l'angle d'inclinaison.

Pour cela nous avons calculé le nombre de pixels noirs sur une ligne de pixels de l'image, fait la moyenne du nombre de pixels par ligne et enfin, calculé la moyenne générale du nombre de pixels par lignes. Par la suite, nous avons calculé la variance du nombre de pixels par ligne. Plus la variance est forte, plus il y a de pixels noirs sur une même ligne

(respectivement de pixels blancs) et donc plus l'image est droite.

Ainsi, tant que la valeur de la variance est supérieure à la précédente valeur, la fonction refait les calculs des moyennes et de la variance après avoir effectué une rotation de l'image vers la droite de 1 ° (si la valeur de la variance est inférieure, elle fera une rotation vers la gauche d'1 °).

Après avoir trouvé l'angle d'inclinaison, nous effectuons la rotation de l'image avec l'angle obtenu grâce à la fonction expliquée dans le paragraphe suivant.

6.2.2 Utilisation de l'angle trouvé

Après avoir trouvé l'angle d'inclinaison, il nous a fallut l'utiliser pour faire tourner notre image.

Pour notre premier essai, nous avons utilisé ces formules pour le calcule des nouvelles coordonnées :

x' = x*cos(angle) – y*sin(angle)
y' = x*sin(angle) +y*cos(angle)

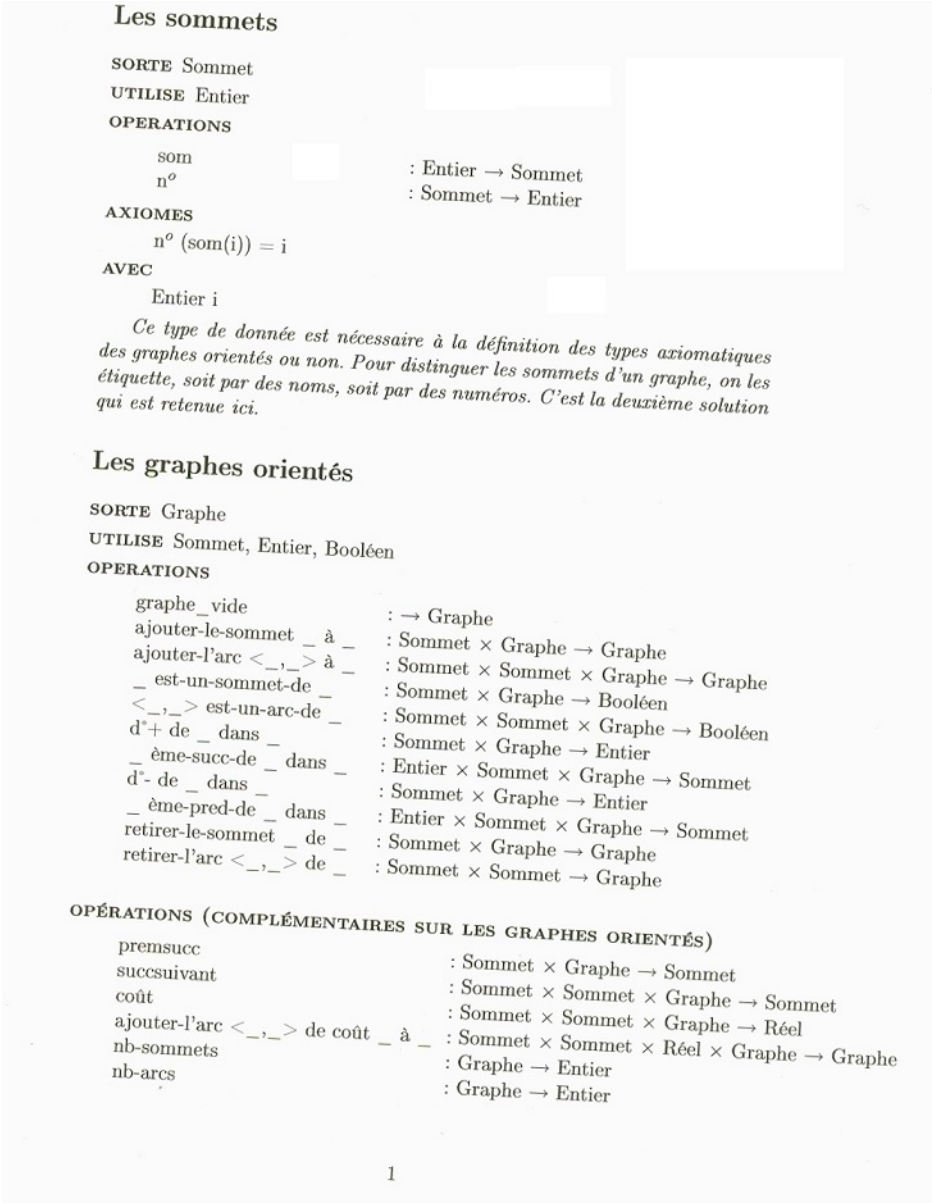
Le problème de ces deux équations est qu'après la transformation, l'image était coupé sur les bords et donc il y avait une grande perte de texte.

Pour résoudre ce problème, nous avons fait des recherches et nous avons trouvé notre erreur fatale : notre rotation débutait sur la coordonnée (0,0) de la matrice (le premier pixel en haut à gauche de la matrice).
Or il fallait débuter par le pixel situé au centre de l'image : en effet, il faut prendre le pixel du centre comme centre de la rotation.
Ainsi les bonnes formules a utiliser sont :

x' = centre + (x – centre) * cos(angle) – (y – centre) * sin(angle)
y' = centre + (x – centre) * sin(angle) + (y – centre) * cos(angle)

6.2.3 Application de la rotation

Voici un exemple de ce que nous avons pu obtenir grâce à la fonction de rotation d'une image :



Les sommets

SORTE Sommet
UTILISE Entier
OPERATIONS

som	: Entier → Sommet
n^o	: Sommet → Entier

AXIOMES

n^o (som(i)) = i

AVEC

Entier i

Ce type de donnée est nécessaire à la définition des types axiomatiques des graphes orientés ou non. Pour distinguer les sommets d'un graphe, on les étiquette, soit par des noms, soit par des numéros. C'est la deuxième solution qui est retenue ici.

Les graphes orientés

SORTE Graphe
UTILISE Sommet, Entier, Booléen
OPERATIONS

graphe_vide	: → Graphe
ajouter-le-sommet _ à _	: Sommet × Graphe → Graphe
ajouter-l'arc <_,_> à _	: Sommet × Sommet × Graphe → Graphe
_ est-un-sommet-de _	: Sommet × Graphe → Booléen
<_,_> est-un-arc-de _	: Sommet × Sommet × Graphe → Booléen
d°+ de _ dans _	: Sommet × Graphe → Entier
_ ème-succ-de _ dans _	: Entier × Sommet × Graphe → Sommet
d°- de _ dans _	: Sommet × Graphe → Entier
_ ème-pred-de _ dans _	: Entier × Sommet × Graphe → Sommet
retirer-le-sommet _ de _	: Sommet × Graphe → Graphe
retirer-l'arc <_,_> de _	: Sommet × Sommet → Graphe

OPÉRATIONS (COMPLÉMENTAIRES SUR LES GRAPHES ORIENTÉS)

premsucc	: Sommet × Graphe → Sommet
succsuivant	: Sommet × Sommet × Graphe → Sommet
coût	: Sommet × Sommet × Graphe → Réel
ajouter-l'arc <_,_> de coût _ à _	: Sommet × Sommet × Réel × Graphe → Graphe
nb-sommets	: Graphe → Entier
nb-arcs	: Graphe → Entier

1

Les sommets

SORTE Sommet

UTILISE Entier

OPERATIONS

som	: Entier → Sommet
n^o	: Sommet → Entier

AXIOMES

n^o (som(i)) = i

AVEC

Entier i

Ce type de donnée est nécessaire à la définition des types axiomatiques des graphes orientés ou non. Pour distinguer les sommets d'un graphe, on les étiquette, soit par des noms, soit par des numéros. C'est la deuxième solution qui est retenue ici.

Les graphes orientés

SORTE Graphe

UTILISE Sommet, Entier, Booléen

OPERATIONS

graphe_vide	: → Graphe
ajouter-le-sommet _ à _	: Sommet × Graphe → Graphe
ajouter-l'arc <_,_> à _	: Sommet × Sommet × Graphe → Graphe
_ est-un-sommet-de _	: Sommet × Graphe → Booléen
<_,_> est-un-arc-de _	: Sommet × Sommet × Graphe → Booléen
d°+ de _ dans _	: Sommet × Graphe → Entier
_ ème-succ-de _ dans _	: Entier × Sommet × Graphe → Sommet
d°- de _ dans _	: Sommet × Graphe → Entier
_ ème-pred-de _ dans _	: Entier × Sommet × Graphe → Sommet
retirer-le-sommet _ de _	: Sommet × Graphe → Graphe
retirer-l'arc <_,_> de _	: Sommet × Sommet → Graphe

OPÉRATIONS (COMPLÉMENTAIRES SUR LES GRAPHES ORIENTÉS)

premsucc	: Sommet × Graphe → Sommet
succsuivant	: Sommet × Sommet × Graphe → Sommet
coût	: Sommet × Sommet × Graphe → Réel
ajouter-l'arc <_,_> de coût _ à _	: Sommet × Sommet × Réel × Graphe → Graphe
nb-sommets	: Graphe → Entier
nb-arcs	: Graphe → Entier

1

CHAPITRE 7

DÉTECTION DES ZONES DE TEXTES

Après avoir "nettoyé" au maximum l'image de toutes ses impuretés, il faut s'attaquer à la localisation des lignes et à la détection des bornes supérieure et inférieure de chaque ligne.

7.1 But

Le but de cette détection des zones de texte est de permettre de localiser chaque caractère d'une phrase afin de les retranscrire dans le document texte final.

7.2 Principe de l'algorithme X/Y cut

Afin de localiser les zones de texte, l'algorithme appelé X/Y cut peut être utilisé. Cet algorithme permet de découper le fichier en plusieurs zones en commençant tout d'abord par les plus grandes surfaces puis en restreignant le découpage dans chaque bloc trouvé jusqu'à ce qu'il n'y ait plus de possibilité de faire d'autres découpages.

Ainsi l'algorithme X/Y cut parcourt l'image d'abord horizontalement et détecte les changements de paragraphes en hauteur (c'est à dire lorsqu'il y a plusieurs lignes composées entièrement de pixels blancs les uns à la suite des autres).

Puis sur chaque sous bloc détecté on applique l'algorithme verticalement pour détecter les paragraphes "parallèles" sur une même hauteur (c'est à dire pour localiser les colonnes de texte).

On recommence tant que l'on continue à détecter des séparations. A la fin on se retrouve avec une liste de blocs et de sous blocs qui correspondent au découpage de l'image.

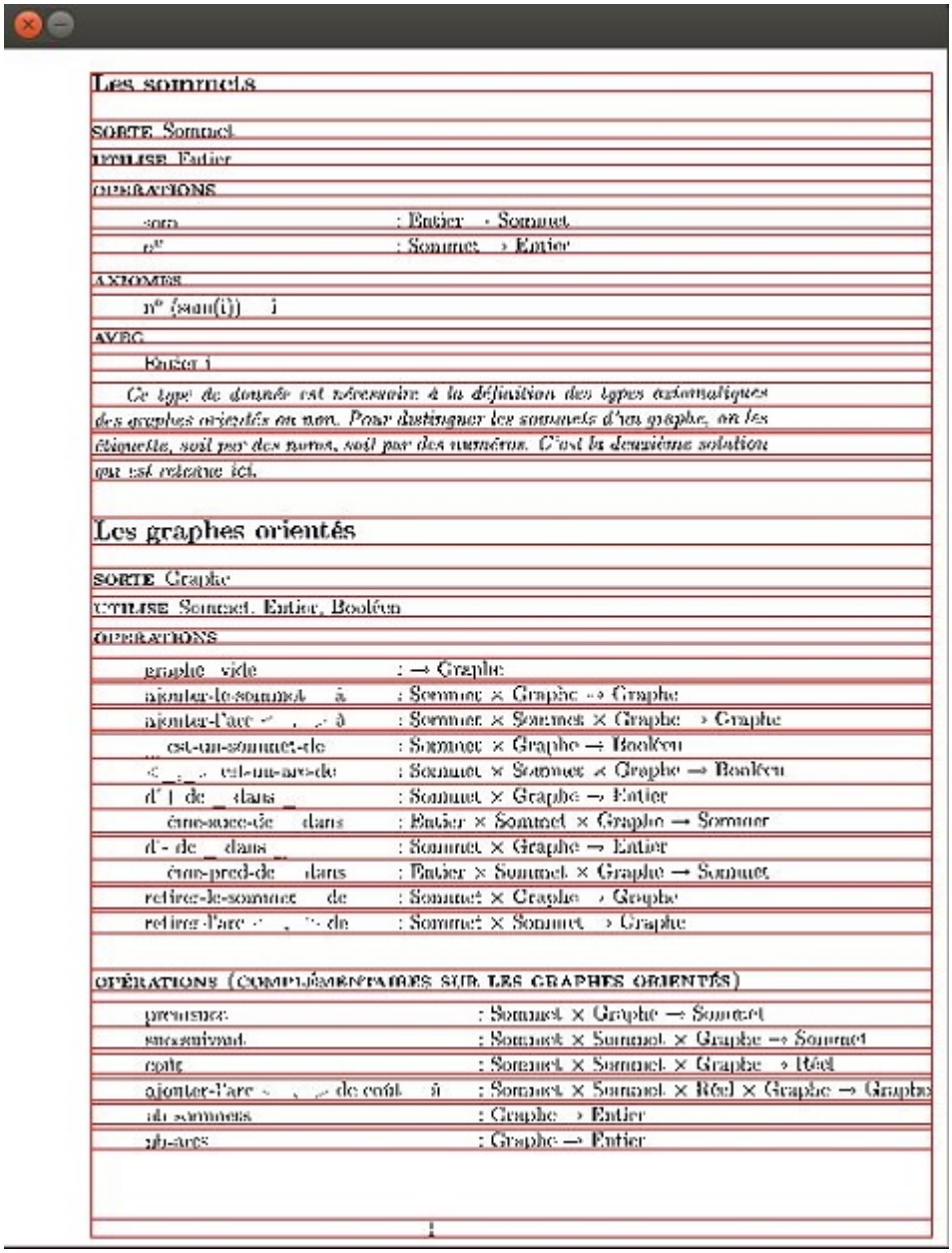
Pour bien faire il faudrait placer les blocs et sous blocs dans un arbre général avec pour racine l'image de départ (sans les pixels blancs autour) et où chaque fils serait le sous bloc détecté par l'algorithme. Cette technique permettrait que les paragraphes soient tous en feuille, ce qui sera plus pratique pour la détection des lignes mais aussi pour savoir dans quel ordre la lecture devra se faire pour retranscrire le texte.

Et pour la détection des lignes à chaque fois que l'on a une ligne de pixel entièrement blanche c'est qu'il y a une nouvelle ligne donc on stocke les coordonnées dans une liste.

7.3 Principe que nous avions implémenté pour la première soutenance

Nous n'avions pas réussi à implémenter l'algorithme X/Y cut. Mais nous avions utilisé un autre principe.

Notre principe était de détecter les lignes et les blocs "en même temps" puis de diviser les blocs trouvés s'ils étaient trop grands afin de faire de nouvelles lignes. Les résultats étaient plutôt bons comme vous pouvez le voir sur cette image :



Les sommets

SORTE Sommet

UTILISE Entier

OPÉRATIONS

som	: Entier → Sommet
n°	: Sommet → Entier

AXIOMES

n°(som(i)) = i

AVEC

i ∈ Entier

Ce type de donnée est nécessaire à la définition des types axiomatiques des graphes orientés ou non. Pour distinguer les sommets d'un graphe, on les étiquette, soit par des noms, soit par des numéros. C'est la deuxième solution qui est retenue ici.

Les graphes orientés

SORTE Graphe

UTILISE Sommet, Entier, Booléen

OPÉRATIONS

graphe vide	: → Graphe
ajouter-le-sommet s à	: Sommet × Graphe → Graphe
ajouter-l'arc s, s' à	: Sommet × Sommet × Graphe → Graphe
s est-un-sommet-de g	: Sommet × Graphe → Booléen
<s, s'> est-un-arc-de g	: Sommet × Sommet × Graphe → Booléen
d°+ de s dans g	: Sommet × Graphe → Entier
i ème-succ-de s dans g	: Entier × Sommet × Graphe → Sommet
d°- de s dans g	: Sommet × Graphe → Entier
i ème-pred-de s dans g	: Entier × Sommet × Graphe → Sommet
retirer-le-sommet s de g	: Sommet × Graphe → Graphe
retirer-l'arc s, s' de g	: Sommet × Sommet × Graphe → Graphe

OPÉRATIONS (COMPLÉMENTAIRES) SUR LES GRAPHES ORIENTÉS

premier-s	: Sommet × Graphe → Sommet
succ-suivant s	: Sommet × Sommet × Graphe → Sommet
cout <s, s'>	: Sommet × Sommet × Graphe → Réel
ajouter-l'arc <s, s'> de coût c à g	: Sommet × Sommet × Réel × Graphe → Graphe
nb-sommets g	: Graphe → Entier
nb-arcs g	: Graphe → Entier

7.4 Principe que nous avons implémenté pour cette soutenance

Pour la reconnaissance des caractères on part de la détection des lignes et on applique récursivement un algorithme similaire sur chaque ligne avec un seuil d'espacement plus faible.

On stock le tout dans des listes de listes de coordonnées pour les position et taille de chaque caractère (la première liste est une liste des lignes (qui sont elles mêmes des liste de caractères)).

CHAPITRE 8

LE RÉSEAU DE NEURONES

Le terme réseau de neurones n'a pas été choisi par hasard par les informaticiens. En effet, le réseau de neurones informatiques est proche du fonctionnement des neurones biologiques.

8.1 Mais qu'est ce que c'est concrètement ?

Le réseau de neurones est la partie la plus importante du projet. Il permet de décrypter c'est à dire de reconnaître les caractères présents dans l'image ayant subit les différents traitements explicités au début de ce rapport.

Plus précisément, un réseau de neurones est composé de plusieurs neurones ayant une fonction d'activation (dépendant des variables d'entrées) permettant à ceux-ci d'être soit actifs (le neurone a une valeur = 1) soit inactifs (le neurone a une valeur = 0) selon leurs rôles. En fonction de l'activation du neurone, celui-ci renvoi une valeur spécifique en sortie.

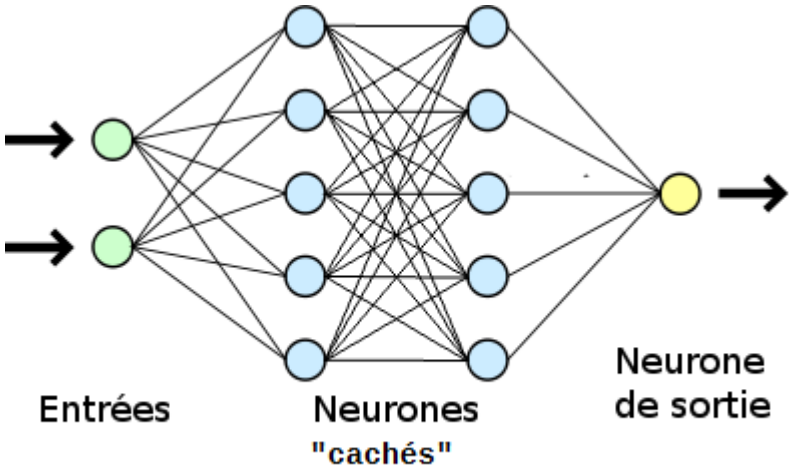
Il existe différents types de réseaux de neurones : le plus répandu étant le perceptron multi-couches nous allons vous l'expliquer en détail.

8.1.1 Le perceptron multi-couches

Un réseau de neurones est composé de plusieurs groupements de neurones. En effet il en faut un pour l'entrée qui représente le caractère isolé que l'on veut retranscrire et un autre pour la sortie qui

identifie le caractère sélectionné. Il existe également des groupements de neurones dits "cachés" qui servent pour les calculs.

Chaque neurone d'un groupe est relié uniquement aux neurones du groupement précédent par des poids. De plus, les neurones d'un même groupe ne sont pas reliés entre eux. Plus précisément, tous les neurones appartenant au groupement de neurones d'entrée vont être reliés à tous les neurones appartenant au groupement "caché" qui eux vont être reliés au groupement de neurones de sortie. Vous comprenez ? ;) Rien ne vaut un schéma explicite :



Nous pouvons préciser que plus il y a de groupements de neurones cachés plus la précision du réseau est grande mais plus il est difficile de l'implémenter...

8.1.2 L'apprentissage

L'apprentissage peut se faire par la méthode de rétro-propagation du gradient, qui réajuste les poids des liaisons neuronales en fonction de tests d'apprentissage (ils sont initialisés avec des valeurs aléatoires). On passe au réseau des entrées d'une valeur choisie (dont on pourra prédire la sortie correcte) et on réajuste en fonction de la sortie (si elle est bonne ou non). Il s'agit de la partie la plus difficile car il faut bien réajuster les poids et faire correctement les tests. C'est cette partie qui

va permettre l'apprentissage qui pourra être sauvegardé pour éviter de le refaire à chaque fois.

8.2 Un semblant de réseau de neurones

Pour cette soutenance, nous n'avons pas réussi à faire "un vrai" réseau de neurones... Ainsi nous avons essayé de contourner notre problème en implémentant un semblant de réseau de neurones mais malheureusement il ne fonctionne pas.... Nous allons tout de même vous expliquer le principe auquel nous avions pensé.

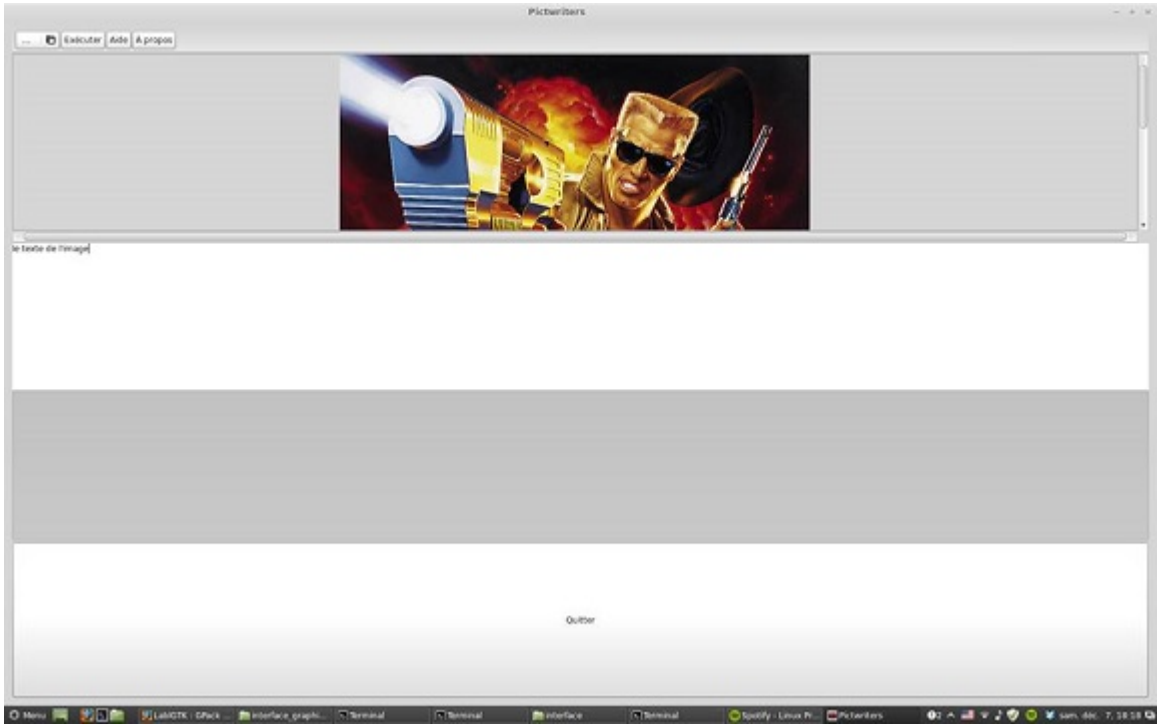
Nous avions pensé faire la reconnaissance en comparant les matrices associées à chaque caractère (on place chaque caractère dans un tableau à deux dimensions avec 0 pour pixel blanc et 1 pour pixel noir).

On comparait ensuite les matrices à un échantillon de lettres connues pour déterminer le caractère (en fonction si ça correspond à 80% ou plus de la matrice "modèle") Mais malheureusement cela ne donne pas de résultats assez satisfaisants.

CHAPITRE 9

L'INTERFACE

Voici notre interface graphique !



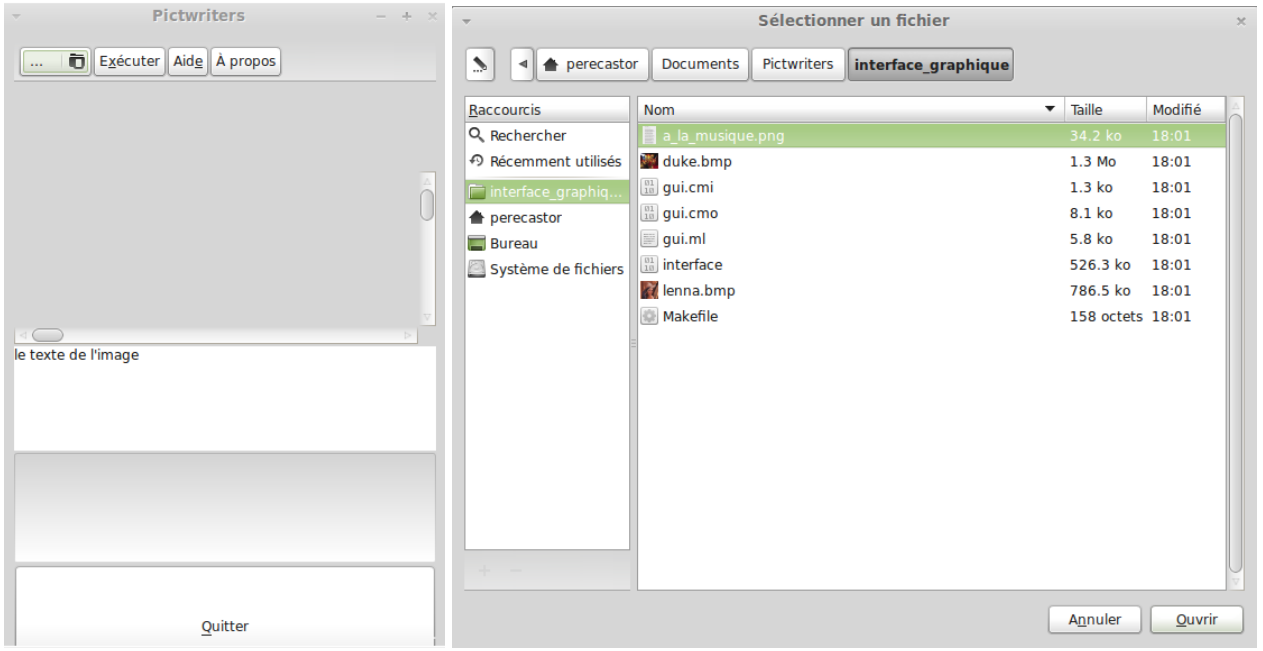
9.1 Son but

L'interface graphique va permettre aux utilisateurs d'utiliser notre OCR d'une manière plus intuitive et donc plus facile.

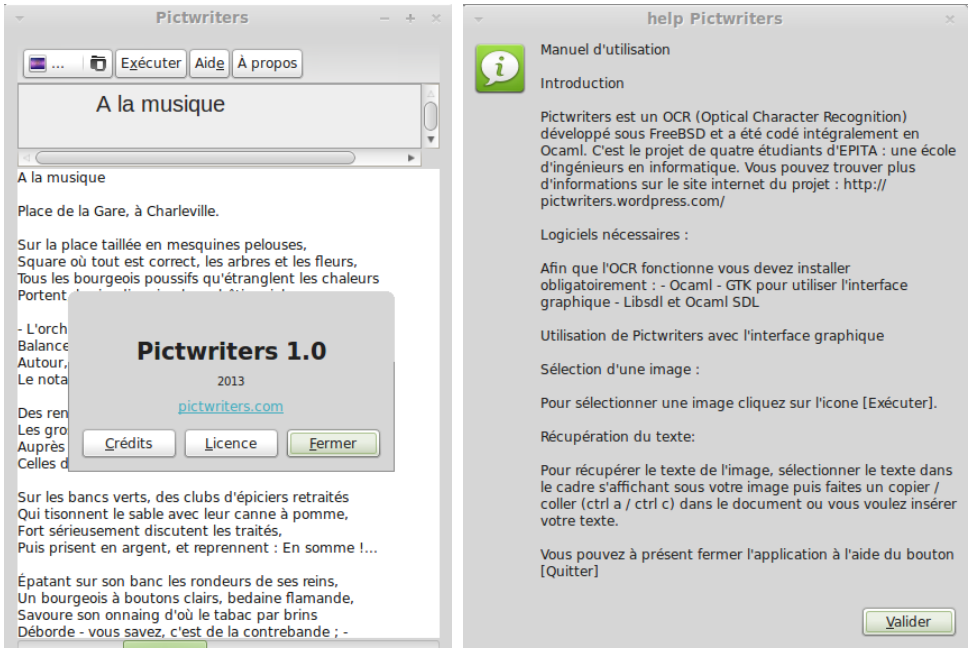
9.2 Description

Nous avons respecté les demandes du cahier des charges qui exigeait que l'interface graphique soit composée d'une fenêtre comportant une zone pour l'image ainsi qu'une zone pour le texte.

Mais ce n'est pas tout!! Notre OCR est composé d'une Toolbar comportant deux boutons principaux : le premier pour sélectionner l'image grâce à un menu simple et intuitif et le second : un bouton "Exécuter" afin de lancer le traitement.



Notre Toolbar est composée de 2 boutons supplémentaires : le bouton "à propos" afin d'obtenir de plus amples informations sur la Team, la licence et permet même d'accéder au site internet officiel, le bouton "Help", quant à lui, ouvre une petite fenêtre avec un manuel d'utilisation simplifié.



Notre OCR possède également une barre de chargement qui permet d'indiquer à l'utilisateur que le traitement est bien en cours.

L'utilisation des GtkStock.id pour tous les boutons de l'interface rend notre interface, en plus d'être **Open Source**, multi-langues.

9.3 Bibliothèques et langages utilisés

Notre interface graphique a été faite en OCaml à l'aide de la bibliothèque GTK. Plus précisément, à l'aide de lablGTK qui est une interface OCaml de GTK+

CHAPITRE 10

NOTRE SITE INTERNET

`http://pictwriters.wordpress.com/`

Voici notre premier site internet ! C'est celui que nous avions fait pour la première soutenance à l'aide de wordpress et que nous avons continué à actualiser pour cette soutenance.

Grâce à celui ci vous pouvez dès à présent accéder au PDF de notre rapport de première soutenance et de seconde soutenance ainsi qu'à la présentation de chacun des membres de la Pictwriters Team ! Nous y avons également intégré notre dossier d'exploitation. Nous avons aussi posté différents articles indiquant certaines informations à propos de notre projet. Vous pouvez cliquer sur un lien afin de voir où nous en sommes dans notre code source (et même de le télécharger) grâce à GIT.

De plus, vous pouvez aller voir les sites internet des éléments utilisés pour faire notre projet grâce à l'onglet liens.

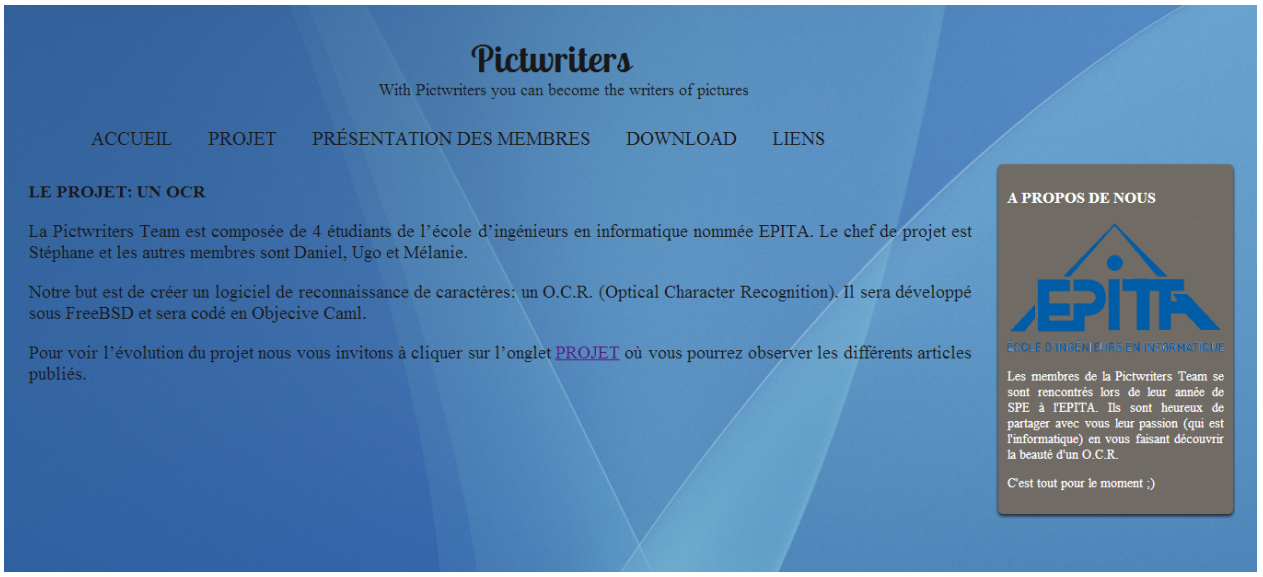
Nous avons également ajouté des "widgets" :
- un calendrier : lorsque l'on clique sur un jour spécial il affiche les articles publiés ce jour même ;
- une liste des articles récents ;
- un décompteur : il permet de calculer le nombre de jours qui sépare le jour actuel à celui d'un événement (pour nous : la soutenance) ;
- un compteur : il compte le nombre de visite sur notre page web

10.1 Finalisation du site internet en langage HTML et CSS

Utiliser wordpress est bien beau mais apprendre à faire son propre site internet est aussi très intéressant. En effet, wordpress est assez facile d'utilisation lorsqu'on commence à faire un site internet alors qu'utiliser le langage HTML et CSS et beaucoup plus complexe et moins intuitif. Cependant, il est véritablement très intéressant de faire son site avec HTML et CSS.

C'est pour cela que nous avons fait un site internet en langage HTML et CSS. Nous avons trouvé une bonne documentation sur le site `http://fr.openclassrooms.com/` (ex. Site du Zéro). Nous avons pu acquérir, grâce à ce site, de nombreuses connaissances en code HTML et CSS.

Voici le résultat de notre code en HTML et CSS :



Nous sommes heureux d'avoir pu héberger notre site internet car il nous tient à cœur.

CHAPITRE 11

DOCUMENTATION

Vous pourrez trouver en plus de notre rapport de soutenance un dossier d'exploitation contenant : un manuel d'installation et un manuel d'utilisation.

Ils ont été intégrés sur le site internet en plus de nos rapports et de la possibilité de récupérer notre code sur Git.

CHAPITRE 12

CONCLUSION

Voici donc que s'annonce la fin de notre aventure de la conception d'un OCR. Notre projet à grandi petit à petit et est devenu ce que vous pouvez voir aujourd'hui : notre bébé à grandi si vite!!

Nous avons seulement eu 3 mois pour concevoir ce logiciel. Ce petit laps de temps nous a quand même permit de vous proposer un logiciel fonctionnel ayant une interface graphique intuitive. Ainsi nous sommes plutôt satisfait de notre travail surtout qu'il nous a permit d'acquérir de nombreuses connaissances en peu de temps : nous avons améliorer notre savoir sur le code en OCaml sous FreeBSD, nous avons appris à faire une interface graphique à l'aide de GTK mais également à faire un site internet en langage HTML et CSS.

Ainsi nous avons pris beaucoup de plaisir à faire ce projet malgré certaines difficultés que nous avons réussies à surmonter.