
# Rapport de projet OCR

## - Présenté par -

SIMONIN "Bictol l'éponge" Victor (Chef de Projet)

OMELKOV "The wizard" Vsevolod

BOFFELLI "Papi Sotiè re" Quentin

CAMINCHER "The whisper" Martin

# Table des matières

# 1 Introduction

Bienvenue dans ce rapport finale du projet OCR ! Dans ce rapport, nous vous présenterons toutes les étapes de la conception de ce projet. L'équipe, composée de Victor SIMONIN, Vsevolod OMELKOV, Quentin BOFFELLI et Martin CAMINCHER, vous présentera l'intégralité de son projet informatique de deuxième année à l'EPITA, qui a été réalisé sur une durée d'environ trois mois.

L'objectif de ce projet était de réaliser un logiciel de type O.C.R (Optical Character Recognition soit Reconnaissance Optique de Caractère) dont la fonctionnalité principale est d'extraire depuis une image le texte contenu dans celle-ci.
L'équipe a au cours de ces dernières semaines principalement travaillé sur l'utilisation et l'apprentissage des syntaxes du langage de programmation C qui est essentiel à la réalisation du projet. Le groupe a travaillé dur pour pouvoir commencer à combler ces exigences, qui représentent un véritable défi, nous obligeant à mener de notre côté de nombreuses recherches personnelles.

Cependant, avec la quantité de travail demandé pendant ce semestre ainsi qu'un départ un peu au ralenti en globalité, ce projet ne s'annonçait pas forcément si facile. Ainsi, l'esprit d'équipe, la communication et la rigueur ont été essentielles pour réussir à avancer dans notre progression, à nous entraider en fonction des niveaux de chacun et à avoir la même vision du rendu de l'OCR.

Ce type de projet est nouveau pour nous, le départ fût difficile, mais chaque membre de l'équipe a donné le meilleur de lui-même afin de faire fonctionner correctement et proprement son projet.

Dans la suite de ce rapport de soutenance final, vous allez parcourir plusieurs parties qui permettent de connaître l'avancement actuel du projet : notamment le DÉCOUPAGE DES TACHES permettant d'aboutir a un OCR, ensuite les différentes RÉALISATIONS de chacun au cours de toute la période de travail et enfin les différentes COMPLICATIONS et RÉUSSITES que nous avons pu rencontrer pendant ce projet, autant au niveau travail qu'au niveau du travail en tant que groupe en général.

## 1.1 SOBC

Le nom du groupe est : The Squad of Overworked but Brilliant Comrades (SOBC).

L'équipe est composée de quatre membres (qui vous seront présentés plus loin). Ses membres se sont connus durant toute la première année à l'EPITA.

Ce groupe est pour le moins atypique et quatre personnalités bien différente se rejoignent pour réaliser avec succès ce projet. Il est important de noter que la cohésion d'un groupe dans un projet de cette envergure est une composante majeure dans sa réussite globale et que le fait de ne pas ou peu se connaître ou de ne pas partager les mêmes horaires peut être un frein dans l'avancement générale ou dans la communication entre les membres.

## 1.2 L'équipe

### 1.2.1 Victor Simonin (Chef de Projet)

Motivé, perspicace et possédant un humour ravageur, ce musicien-skieur Savoyard est un atout inestimable dans cette équipe. Victor surnommé "Bictole l'éponge" est remonté comme jamais pour ce projet et pour le mener à bien jusqu'au bout en tenant les rênes de son équipe même s'il faut faire des sacrifices pour cela. Il faut néanmoins surveiller son aptitude aux erreurs d'allocation mémoire et son talent sans équivoque pour la mise en feu de son ordinateur.

### 1.2.2 Vsevolod Omelkov

Le seul étudiant russe de toute l'EPITA, il travaille sur le réseau de neurones et son fonctionnement. Son surnom est "The wizard" car son job est de créer LA VIE... Ou du moins quelque chose qui lui ressemble en forme d'une émulation du cerveau humain qui est le réseau de neurones.

### 1.2.3 Quentin Boffelli

Originaire de Grenoble, ce jeune étudiant d'EPITA, bourré .... d'humour est bien décidé a faire ce projet jusqu'à la fin. En effet grâce a ce dernier l'équipe pourra approcher de très prés ce qui ce fait en entreprise, ce qui est quelque chose dont il raffole.

### 1.2.4 Martin Camincher

Cet étudiant originaire de la campagne, assez calme et silencieux, ce qui lui a valu le surnom de "The whisper", est décidé a donner tout ce qu'il a afin de mener ce projet à terme. Bien que les projets en groupes ne sont pas son domaine de prédilection, il compte bien ne pas laisser cela le gêner dans la réalisation de ce projet.

# 2 Découpage du projet

## 2.1 Chargement de l'image et suppression des couleurs

Cette partie n'est pas très complexe, mais est essentielle au reste du projet. Il s'agit de fournir au reste du programme une image, et de la modifier afin que celle-ci soit plus adaptée au traitements qui lui seront ensuite effectués. La suppression des couleurs permet de nettoyer l'image de toutes ses impuretés. En effet, même une image en noir et blanc normale contient très souvent des nuances de gris. En la passant en noir et blanc pur, l'algorithme n'a plus qu'a s'occuper uniquement de deux types de pixels, noir ou blanc, ce qui rend le traitement bien plus aisé.

## 2.2 Pré-traitement

Le pré-traitement n'est pas forcément nécessaire au fonctionnement du projet, mais il permet d'avoir une plus grande polyvalence au niveau de l'entrée utilisateur, c'est à dire au niveau des images données au logiciel. De nombreuses idées de pré-traitement peuvent venir à l'esprit assez rapidement comme par exemple enlever certaines taches présentes sur le document à traiter ou modifier la rotation d'une image qui ne serait pas tout à fait droite et plus encore. Cette partie peut, comme d'autres dans ce projet, être indéfiniment amélioré pour toujours plus de perfection et de précision.
Le pré-traitement est bonus et ne sera implémenté dans l'OCR dans le seul cas ou toutes les autres parties seront opérationnelles et optimisées.

## 2.3 Découpage de l'image

Le découpage de l'image est une partie technique et essentielle pour le bon fonctionnement de l'OCR. Elle consiste dans les grandes lignes à prendre l'image dans sa représentation binarisé, de la découper selon les caractères présents, puis d'envoyer ces caractères dans un ordre précis au réseau de neurones qui se chargera de les reconnaître. Il est important de bien lier cette méthode avec la reconstruction du texte, en effet il faut se rappeler quels caractères se place à quels endroits pour pouvoir reconstruire le texte rapidement

De nombreux algorithmes de découpage existent déjà. Ils sont plus ou moins facile à implémenter et plus ou moins efficace selon les images qu'on leur donne.

## 2.4 Reconnaissance de caractères : réseau de neurones

Un réseau de neurones est, en effet, un réseau, c'est-à-dire une collection d'éléments liés entre eux, de neurones, ou plutôt, des émulations mathématiques des connections synaptiques se trouvant dans un cerveau humain. Comme dit précédemment, la première étape de la construction d'un réseau de neurones (ultérieurement abrévié comme RN) est la création du réseau en lui même, qui consiste, essentiellement, à définir des "traits" de notre problème a résoudre, c'est-à-dire, déterminer le nombre de neurones nécessaire et de les donner une fonction, qui leur permettra de réagir d'une certaine façon si un certain élément est présent dans la fonction analysée. Avant de continuer, définissons comment un neurone marche. En effet, c'est assez simple : on a des conditions d'entrée, chacune possédant un poids (son importance pour l'activation du neurone) et un certain seuil qui est propre a chaque neurone. Si une condition d'entrée est vérifié, on ajoute son poids avec les autres poids des conditions vérifiés, et on regarde si on a franchit le seuil du neurone, et si ce dernier est vrai, le neurone est activé. C'est effectivement tout simple ! D'ailleurs, on utilise des neurones sigmoïdes, qui retournent des valeurs des valeurs entre 0 et 1, mais on pourrait en penser comme si c'était des pourcentages de probabilité que la condition d'entrée est vraie.

Puis, on va juste chaîner les neurones entre eux et initialiser les poids de façon aléatoire (cela est possible car le RN va modifier les poids lors de l'apprentissage) et on a notre RN qui est prêt pour apprendre ! Mais comment peut-il apprendre ?

Bah, cela est encore plus facile que le fonctionnement des neurones : on va lui donner une base de données, dont on connais déjà la réponse a chaque unité d'apprentissage, et on lui fera travailler avec toutes les unités de la base de données, un par un (ou il peut travaille dans ce qu'on appelle des "batch", mais cela consiste a simplement faire une approximation du résultat des multiples unités pris ensemble) et puis comparer ces résultats avec ceux attendus. Ce processus s'appelle "Feed Forward", et c'est lui qui sera utilisé pour la mise en marche du RN après. Cependant, lors de l'apprentissage, on a un autre processus très important qui s'effectue juste après le "Forward Pass" : la "Backpropagation", propagation inversé. Ce processus est ce que fait que notre RN peut apprendre á partir de ces erreurs. Ce qu'il fait, essentiellement, est d'ajuster les poids et les biais (les seuils) de ces neurones en relation avec la validité du résultat du "Feed Forward" comparé avec les résultats attendus. Ainsi, on a un RN qui peut apprendre à partir de ces erreurs, tout comme un bon étudiant d'EPITA !

Passons ainsi à la mise en marche de notre RN : cela est encore plus simple que les parties d'avant. Tous ce qu'on a à faire, après qu'un RN a été bien entraîné, est de lui passer en paramètre ce qu'on veut qu'il analyse et puis retirer le résultat. Ce n'est pas plus difficile que ça. Il existe un problème important a gérer en réseaux de neurones : le surentraînement. Cela consiste essentiellement au fait que le réseau a passé trop du temps avec la base de données du départ et son efficacité est beaucoup moins présente que s'il se serait entraîné pendant moins du temps. D'où, la création des réseaux de neurones est souvent appelée un art plus qu'une science, car chaque cas d'application des réseaux de neurones nécessite une approche personnelle et une construction minutieuse.

## 2.5 Reconstruction du texte

La reconstruction du texte est une section qui dépend énormément du découpage de l'image et de la segmentation des caractères. En effet la méthode utilisé pour extraire les caractères de l'image va déterminer la méthode pour reconstruire le texte. Cette branche de l'OCR consiste donc à reprendre les caractères reconnus par le réseau de neurones et à les assembler sur un document texte le plus fidèlement possible à l'image originale en respectant donc bien les espaces, les lignes et les paragraphes. La reconstruction est une étape algorithmique un peu technique et nécessite de la rigueur et de l'attention.

## 2.6 Interface Graphique

L'interface graphique est une partie assez éloigné des autres dans ce projet d'OCR. En effet assez loin de l'algorithmique pure, de la gestion d'image ou du réseau de neurone, l'interface nécessite d'être fonctionnelle, pratique et même d'avoir un joli rendu graphique.
Cette section ne sera réalisé qu'en deuxième partie de notre OCR entre la première et la seconde soutenance lorsque la majorité des composantes majeures de notre logiciel final seront implémentées.
Elle devra globalement permettre à l'utilisateur de charger son image, d'effectuer facilement notre programme de reconnaissance de caractères et enfin de récupérer son texte final en le sauvegardant sur son ordinateur.

Loin d'être facile, l'interface graphique va nécessiter de la rigueur et de la réflexion pour être optimisée et agréable à utiliser. En effet, c'est le seul contact qui est fait avec l'utilisateur et il est important que celui-ci soit le plus efficace et soigné possible.

## 2.7 Répartition des tâches et planning d'avancement

| | **Responsable** | **Suppléant** |
|---|---|---|
| **Pré-Traitement** | Quentin | / |
| **Chargement Image/Suppression des couleurs** | Martin | / |
| **Découpage du texte, Segmentation** | Victor | / |
| **Réseau de neurone, XOR** | Vsevolod | / |
| **Reconstruction du texte** | Victor | / |
| **Interface Graphique** | Martin | / |

**Répartition des tâches !**

| | **1ère Soutenance** | **Soutenance Finale** |
|---|---|---|
| **Pré-Traitement** | 0% | 100% |
| **Chargement Image/Suppression des couleurs** | 90% | 100% |
| **Découpage du texte, Segmentation** | 70% | 99,99999% |
| **Réseau de neurone, XOR** | 50% | 100% |
| **Reconstruction du texte** | 0% | 100% |
| **Interface Graphique** | 0% | 100% |

**Planning d'Avancement**

# 3 Conception

## 3.1 Chargement de l'image et suppression des couleurs

Le chargement de l'image ainsi que la suppression des couleurs est une partie essentielle du projet. En effet, celle-ci permet au programme d'obtenir l'image qu'il utilisera, et de la transformer en un format plus adapté au traitements qui suivront.

Martin :

Cette partie fut retravaillée après la 1ère soutenance, et a beaucoup évoluée. En effet, l'image est à présent chargée dans dans la partie du programme qui gère l'interface graphique. Une fois que le traitement de l'image commence, celle-ci est envoyée au programme Black_White, qui a pour objectif de transformer l'image en une matrice de 0 et de 1 représentants les pixels noirs et blancs. L'image obtenue en paramètre est donc parcourue deux fois. La première fois, chaque pixel est observé et on calcul sa couleur en nuance de gris. la valeur d'un pixel en nuance de gris est définie comme étant 30% de sa valeur de rouge, plus 59% de sa valeur de vert, plus 11% de sa valeur de bleu. Ce calcul est plus utile qu'une simple moyenne des valeurs des trois couleurs car les différentes couleurs élémentaires rouge, vert et bleu sont perçues différemment par notre oeil, et doivent donc passer par un tel calcul afin de donner la valeur de gris la plus proche possible de la couleur originale du pixel.

On fait ensuite la moyenne des valeurs de gris de l'image, et on obtient la couleur moyenne de l'image. Si on assume que l'image est constituée de deux couleurs différentes, pour le texte et pour le fond, il est ainsi plus facile de les séparer et donc de pouvoir identifier le texte dans l'image, plutôt que d'utiliser la valeur $255/2 = 128$.

Après avoir calculé cette moyenne, on parcours à nouveau l'image et on place dans une matrice de même dimension que l'image les valeurs 0 ou 1 en fonction de si la valeur de gris du pixel correspondant est supérieur ou inférieur à la couleur moyenne de l'image.

On ne renvoie ensuite pas cette matrice, celle-ci récupérant depuis le programme principal où elle a été initialisée tous les changements qui lui sont effectués dans le programmes secondaire.

Cette matrice représentant l'image est donc prête à être ensuite envoyée au reste de l'algorithme.

## 3.2 Binarisation de l image

La binarisation permet de passer d'une image avec des pixels a un tableau de valeur contenant uniquement des 1 et des 0. Vous l'aurez sans doutes compris, cette partie est essentielle au bon fonctionnement de l'OCR.

Quentin :

J'ai travaillé sur la binarisation et notamment sur la méthode d Otsu. En effet après l erreur de la première soutenance, j'ai decidé de rattraper ça. Pour rappel la methode d'Otsu permet d avoir une binarisation plus "propre". Cette fonction de binarisation prend en parametre une image sous format sdl et renvoie une image en noir et blanc (binarisé).

## 3.3 Découpage de l'image

Le découpage de l'image en différents blocs puis en caractères est nécessaire pour envoyer à la reconnaissance les morceaux d'images correspondants logiquement à des caractères.

Victor :

Pour ma part, je me suis occupé intégralement de cette partie pendant toute la période de travail. Cela a commencé avec un travail important et nécessaire de recherche pour savoir quelle destination donner à mes algorithmes et à mon code en général.
Deux algorithmes bien distincts sont alors ressortis, le premier appelé vulgairement le XY-Cut et le second l'algorithme de RLSA (Run Length Smoothing Algorithm).

J'ai tout d'abord assez rapidement implémenté le premier, le XY-Cut. Cette méthode assez naïve consiste a parcourir l'image en noir et blanc ou sa représentation en matrice binarisé et à relever les endroits où se situent des pixels noir et donc des paragraphes, lignes et caractères. J'ai donc réalisé cet algorithme qui me permet d'abord de façon horizontale de relever dans un tableau les coordonnées de mes lignes, puis de relancer cette méthode de façon verticale dans chaque ligne afin de récupérer les coordonnées de chacun de mes caractères sur mon image.

Cet algorithme est assez coûteux au niveau de sa complexité et de son temps de calcul et il est difficile de récupérer les paragraphes et les espaces important sans d'autres tests également coûteux. Je prévois néanmoins d'utiliser cette méthode pour récupérer mes caractères une fois les paragraphes et lignes pré-découpés par mon second algorithme : le RLSA.

Après ce premier algorithme, j'ai décidé de me renseigner un peu plus sur ce second algorithme assez efficace appelée RLSA. Celui-ci est destiné à être appliqué horizontalement et verticalement puis de faire une opération logique sur le résultat de chacun.

En fait cela consiste réellement a prendre tous les pixels blancs qui sont entre deux pixels noirs, et à les modifier pour les passer eux aussi en noir. Une fois appliqué horizontalement et verticalement, il suffit de réaliser un ET logique entre les deux images après la première partie, c'est à dire garder en noir tous les pixels qui sont noirs sur la première image (après l'application verticale) ET sur la seconde (après l'application horizontale). Cela nous donne une troisième image qui est donc la représentation fini de l'algorithme de RLSA [1].
Pour obtenir un résultat plus propre après cela, j'ai décidé de relancer mon RLSA de façon horizontale et verticale, ce qui me permet de ressortir avec des blocs facile a traiter par la suite.

Cet algorithme est très efficace et permet de reconnaître les blocs de texte avec une très bonne probabilité de réussite. Néanmoins, il nécessite une valeur assez difficile à calculer, la valeur de l'écart accepté entre deux pixels noirs. J'ai pendant une bonne partie du projet avancé avec des valeurs génériques qui fonctionnaient dans la plupart des textes utilisés pour les tests de mes algorithmes mais même si cela présentait des résultats assez correct dans de nombreux cas, je me suis pendant un assez long moment sur une façon de calculer cette valeur de façon assez optimisé pour chaque image.

Une autre problématique assez importante est également apparu assez rapidement durant la conception de cette segmentation. Ma première méthode de découpage fonctionnait de façon très optimisé sur des textes ayant des caractères au moins séparé d'un pixel blanc et donc pas sur des caractères qui seraient collés. J'ai donc implémenté une nouvelle méthode pour essayer de récupérer également les caractères collés. Cette nouvelle méthode, qui fonctionne avec une proportion du nombre de pixels noirs dans l'axe verticale des caractères, m'a donné des résultats assez peu satisfaisant avec de nombreuses erreurs et cela même avec des caractères espacés.

Par la suite, j'ai pu mettre en place l'algorithme final en liant les forces de mes deux méthodes déjà fonctionnelles et en mettant en place une façon de stocker les données qui me seront utiles pour la reconstruction.

1. Voir Annexe

Le principe de mon algorithme final est donc le suivant :

- **Récupération des paragraphes** : Applique l'algorithme de RLSA plusieurs fois puis l'algorithme de XY-Cut pour sur mon image pour récupérer les paragraphes :
  - **RLSA Horizontalement**
  - **RLSA Verticalement**
  - **Opérateur Logique ET** : L'image en sortie est ici très facile à traiter.
  - **Application du XY-Cut Horizontalement**
  - **Récupération et Stockage des coordonnées des paragraphes sur mon image**
- **Récupération des lignes** : Applique l'algorithme de RLSA une seule fois puis l'algorithme de XY-Cut pour sur mon image pour récupérer les lignes :
  - **RLSA Horizontalement**
  - **Application du XY-Cut Horizontalement**
  - **Récupération et Stockage des coordonnées des lignes sur mon image**
- **Récupération des mots** : Applique l'algorithme de RLSA une seule fois puis l'algorithme de XY-Cut pour sur mon image pour récupérer les mots :
  - **RLSA Horizontalement**
  - **Application du XY-Cut Verticalement sur chaque ligne**
  - **Récupération et Stockage des coordonnées des mots sur mon image**
- **Récupération des caractères** : Applique l'algorithme de XY-Cut sur mon image pour récupérer les mots :
  - **Application du XY-Cut Verticalement sur chaque ligne**
  - **Récupération et Stockage des coordonnées des caractères sur mon image**

Pour la réussite de ce travail, j'ai également du programmer différentes fonctions notamment une fonction de mise en noir et blanc d'une image plutôt optimisé pour ma segmentation ainsi qu'une une fonction d'écriture de matrice pour mes tests.

Pour conclure, il faut mettre de l'importance sur le fait que la segmentation et le découpage du texte peuvent être toujours amélioré, pour fonctionner sur des textes manuscrits, avec des blocs de texte imbriqué dans les autres ou tout autre cas qui peuvent exister.

## 3.4 Reconstruction du texte

La reconstruction du texte est une étape en lien direct avec la segmentation et le découpage du texte. En effet, cette partie dépend intégralement de la façon dont l'image a été traité auparavant et de la façon dont les données, comme les différents espaces ou les retours à la ligne, ont été stockées.

Victor :

Étant donné que c'est moi qui me suis occupé intégralement de la segmentation et du découpage du texte, je me suis attelé de façon assez naturelle à sa reconstruction.

J'ai donc utilisé les données que j'avais stocké auparavant pour reconstruire le plus fidèlement possible le texte rencontré sur l'image. Ayant déjà récupéré les coordonnées de tout mes différents ensembles qui sont les paragraphes, les lignes, les mots et les caractères, il m'a fallu assez simplement réorganisé toutes ces données pour pouvoir replacer au bonne endroit les différents espaces, retour à la ligne ou même saut de ligne. En une fonction et différents tests de coordonnées pour savoir un peu plus précisément où je me situais, j'ai pu assez rapidement produire une reconstruction viable bien avant que la reconnaissance de caractères ne soit mise en place.

J'avais au départ estimé que la représentation de mes bases de données sous forme d'arbre serait le plus optimisé et le plus adapté à un projet comme celui-ci. J'ai donc tout d'abord implémenté cette structure. Je suis parti sur une implémentation d'arbre vu cette année en algorithmique qui sont les arbres généraux premier-fils frère-droit car la structure globale est assez simple à réaliser et la complexité est correcte. J'ai également pu écrire une première reconstruction de cette arbre grâce à un parcours en profondeur assez classique. Cette première idée d'arbre était composé de l'image en racine, puis de façon récursive avec les paragraphes en fils, puis les lignes, mots et caractères.

Peu après, j'ai finalement convenu que je n'utiliserais pas cette structure ni ce type de stockage de données pour reconstruire mon texte. Étant donné que je ne stocke pas de matrice complète mais seulement des coordonnées, je ne pensais pas avoir besoin d'une implémentation aussi complexe. Le fait que je stockais déjà mes coordonnées dans différents tableaux lors de ma segmentation à définitivement fini de me convaincre de partir sur une autre voie, sûrement plus simple et dont le stockage en mémoire et la complexité sont environ équivalents. L'algorithme final reconstruit donc de la façon la plus fidèle possible mon texte à partir de ces tableaux de coordonnées.

À la fin de cette étape de reconstruction, j'ai pu me pencher sur l'écriture et la sauvegarde du texte final dans un document directement dans l'ordinateur de l'utilisateur. La documentation sur ce sujet est assez clair et m'a permis de réaliser cet algorithme sans accrocs [2]. Un document texte peux donc être très facilement récupéré par l'utilisateur directement dans le dossier courant. Le cas où il existerait déjà un fichier du même nom que celui que l'on veut créer est bien géré car mon algorithme viderai alors ce fichier pour le remplir avec le texte final.

2. Voir Annexe

## 3.5 Reconnaissance de caractères : réseau de neurone

Le principe du réseau de neurones est de simuler les neurones humains pour en faire une structure mathématique capable de traiter des données de tous types. Il faut distinguer 3 grandes parties d'un réseau de neurones : la création, l'apprentissage et la mise en marche. Pour la première soutenance, nous n'avons qu'a montrer le principe de fonctionnement d'un réseau de neurones, ce qui nous permet de confirmer que l'on comprend le principe de fonctionnement d'un réseau de neurones.

Vsevolod :

Il y a eu quelques difficultés à surmonter pour produire le réseau neuronal final, à savoir, la mise en œuvre de la fonction SoftMax (son rôle est essentiellement de nous donner le résultat final le plus probable à partir de l'analyse matricielle), la création d'un mécanisme de sauvegarde des biais et poids du réseau et, bien sûr, la formation du réseau lui-même.

Pour être exact, la fonction SoftMax est un moyen de normaliser les valeurs d'un ensemble donné pour ainsi obtenir l'indice de la plus grande valeur de l'ensemble, qui est donc considéré comme le "résultat" de l'application de la fonction. Ceci, lorsqu'il est pris dans le cadre du réseau neuronal, est interprété comme la sortie courante du réseau, ce qui nous permet d'avoir une certaine sortie pour une entrée donnée. Dans le cadre de notre projet, la fonction SoftMax, a pour seul but d'analyser les résultats situés sur notre couche de sortie.

La question principale que l'on doit se poser lorsqu'on traite d'un ensemble de données tel qu'une séquence de caractères, est "Comment puis-je le rendre compréhensible pour un ordinateur ? La réponse, dans notre cas, est de transcrire l'image d'un caractère que nous avons dans une matrice de 0 et 1, correspondant respectivement aux pixels blancs et noirs. Ainsi, nous recevons quelque chose qui peut être passé comme argument à notre réseau.

Par conséquent, après avoir donné au réseau les paramètres de la matrice, le nombre de neu-

rones de la couche cachée et le nombre de sorties attendues, nous pouvons raisonnablement nous attendre à ce qu'il agisse conformément à son comportement déjà entraîné, en fournissant les réponses dans les limites de ses capacités. Il est à noter qu'en jouant avec le taux d'apprentissage et le nombre de neurones de la couche cachée, on peut obtenir des résultats plus précis pour certaines valeurs, via le processus d'étalonnage.

En ce qui concerne la sauvegarde et la restauration du réseau de/vers un fichier, nous utilisons simplement la bibliothèque stdio (Standard IN/OUT), et les fonctions "open", "scan" et "fprintf", qui nous permettent de lire ou écrire dans un fichier donné. Il y avait deux problèmes à résoudre : le cas de l'écriture/rétablissement du biais/de la pondération et le cas des nombres en virgule flottante. Le premier cas peut être résolu par une simple itération des données de la matrice pour la partie écriture et la création d'une nouvelle matrice et un remplissage itératif de la partie lecture. Le second cas peut être résolu en assignant un nombre de décimales après le virgule flottante à prendre en compte, qui, pour ce projet, a été fixé à 6.///

Aucune modification du "backprop" ou du "feedforward" n'est nécessaire, car ils restent essentiellement les mêmes que XOR, sans modification des traitements effectués. Par conséquent, nous ne sommes pas obligés de modifier la cage structurelle de notre réseau, ce qui facilite quelque peu l'exécution. Pour séparer la phase de formation et la phase d'exécution, nous créons une fonction séparée à appeler pendant la formation (SGD) et une autre à appeler pendant l'exécution (feedforward)./// Une chose importante que j'ai mise en œuvre dans le temps entre les deux rapports a été de faire une structure matricielle, capable d'être utilisé à l'intérieur de notre réseau neuronal pour stocker les différentes variables, telles que les poids ou les biais. Il s'agit essentiellement d'une structure, définie comme type "matrice", où les lignes/hauteur et colonnes/largeur sont stockées comme "int", définissant les dimensions de la matrice, formatant ainsi le tableau dynamiquement alloué nommé "données" qui contient les valeurs de la matrice, déployé de manière linéaire afin de limiter l'impact de la matrice sur la mémoire et accélérer l'exécution du traitement neural net.///

## 3.6 Constitution de la base de donnée pour le réseau de neurone

Victor :

Dès la première soutenance, j'ai également pu constituer une bonne partie de la base de données pour le réseau de neurone. En effet la phase d'apprentissage du réseau de neurone est plus qu'essentielle a son bon fonctionnement. Il m'a donc fallu récolter différentes images des caractères, en majuscule et en minuscule mais également sous différentes police, que l'on veux reconnaître, en l'occurrence l'alphabet Latin.

J'ai alors décidé de me pencher sur quatre polices très fréquemment utilisé aujourd'hui qui sont les suivantes : Times New Roman, Calibri, Arial et Comic Sans Ms.

## 3.7 Interface graphique

Martin :

Après la première soutenance, je me suis concentré sur le développement de l'interface graphique. Cette partie du projet est assez différente des autres, mais reste essentielle, l'interface étant le seul contact entre l'utilisateur et le programme.

J'ai dans un premier temps du choisir qu'est ce que j'utiliserais pour créer cette interface graphique. J'aurais pus utiliser GTK, que nous avons découvert lors du TP 06, mais j'avais déjà travaillé sur l'interface avant de commencer ce TP, et je ne voulais pas perdre tous le travail que j'avais effectué en changeant de bibliothèque aussi tard. Mon premier choix fut donc d'utiliser la SDL, que nous avions découvert lors du TP 03.

Cependant, après quelques recherches, j'ai découvert l'existence de la SDL 2.0. J'ai donc choisi d'utiliser celle-ci pour créer l'interface graphique. les raisons de ce choix sont que la SDL 2.0 est un version plus récente et retravaillée de la SDL 1.2, dont la dernière version date de janvier 2012. Mais la principale raison pour ce choix est que je trouvais bien plus aidé d'obtenir de la documentation et de l'aide en ligne pour la SDL 2.0, me facilitant ainsi la tache.

J'ai donc du commencer par me renseigner sur le fonctionnement de la SDL 2.0. Beaucoup de fonctionnalités de la SDL 2.0 sont similaires à celles de la SDL 1.2, étant une version plus avancée de la SDL plutot qu'une bibliothèque totalement différente. Cependant, je n'avais pas non plus de très grandes connaissances sur la SDL, ayant simplement été introduit à la gestion d'image lors du TP 03, et n'ayant jamais utilisé la gestion de fenêtres, ce qui serait l'outil que j'utiliserai principalement pour créer mon interface graphique. Il a donc fallu, dans un premier temps, que je recherche et que j'apprenne, et surtout que je comprennes, comment la gestion de fenêtres fonctionne.

Le fonctionnement de l'interface n'est pas très complexe à comprendre, mais sa réalisation m'a longtemps occupé, étant un sujet totalement nouveau et complexe pour moi.

Je vais commencer par expliquer le fonctionnement de ma fonction "pause", que j'utilise tout au long de l'algorithme de l'interface. Cette fonction prend en paramètre une liste de SDL_Keycode. les SDL_Keycode sont une structure de la SDL permettant d'identifier les différentes touches du clavier. Ma fonction prend donc une liste de SDL_Keycode en paramètre, puis attend un SDL_Event en utilisant la fonction SDL_WaitEvent. Un SDL_Event est n'importe quel évènement déclenché pour l'utilisateur, et la fonction SDL_WaitEvent permet, comme son nom l'indique, d'attendre un de ces évènements.
Une fois qu'un évènement s'est produit, si il s'agit d'un SDL_Event de type SDL_KEYDOWN, un type d'évènement correspondant à quand une touche du clavier est pressée, alors la fonction observe chacun des SDL_Keycode qu'elle a obtenue en paramètre, et les compare à celui de la touche qui vient d'être pressée. Si il s'agit bien d'une des touches passée en paramètre, alors la fonction arrête d'attendre et renvoie la touche qui a été pressée. Cette fonction me permet donc, à certains moments, d'attendre que l'utilisateur appuie sur un touche, et savoir la touche retournée permet ainsi d'agir en fonction de la volonté de l'utilisateur. C'est donc cette fonction qui est responsable du lien entre l'utilisateur et le programme.

Je vais maintenant expliquer le fonctionnement du programme de l'interface graphique (1). On commence par créer les listes qui sont passées en paramètre à la fonction "pause", puis la fonction SDL_Init est appelée afin d'initialiser la SDL. La fonction IMG_Init est également appelée. Cela permettra de charger des images utilisant les formats jpg, png et tif, en plus du format bmp qui est initialement géré par la SDL. La fenêtre principale est ensuite générée, et elle nous servira pour toute l'interface graphique. On la nomme donc "OCR". On charge ensuite l'image servant d'écran d'accueil dans une SDL_Surface (2). En effet, n'ayant ni le temps ni le besoin d'avoir la possibilité d'écrire du texte dans l'interface depuis le programme, j'ai décidé que toutes les informations données à l'utilisateur sous forme de texte dans l'interface serait ainsi directement ajoutées à des images qui seront changées au cours de la progression

de l'utilisateur dans l'interface. Le renderer est également créé, ainsi que la texture contenant l'image de l'écran d'accueil. le renderer est ensuite affichée dans la fenêtre, et montre donc à l'utilisateur l'écran d'accueil. Celui-ci montre le nom des membres du groupe, et demande à l'utilisateur d'appuyer sur la touche "espace". La fonction "pause" est donc appelée et attend que l'utilisateur appuie sur la touche "espace".

Celui-ci peut également appuyer sur la touche "échap" afin de fermer la fenêtre et quitter le programme. J'ai en effet fais de sorte à ce que l'utilisateur puisse choisir de quitter le programme à n'importe quel moment, en appuyant sur cette touche.

Après que l'utilisateur ait appuyé sur la touche "espace", on se débarrasse de la surface et de la texture maintenant inutiles, puis on les remplace pour afficher la prochaine image. On demande maintenant à l'utilisateur quel est le format de l'image qu'il souhaite utiliser (3). Cela permet ainsi de prévenir à l'avance l'utilisateur des différents formats d'image que l'algorithme supporte. En effet, l'algorithme ne peut pas charger des images dans tous les formats existants, la bibliothèque SDL_image est déjà utilisée afin de supporter les formats jpg, png et tif, ce qui est très utiles sachant que les formats jpg et png sont bien plus répandus que le format bmp que la SDL utilise par défaut. Cependant, tous les formats ne sont pas supportés, et cela prévient l'utilisateur qu'il ne peut pas utiliser de format pdf par exemple. Après que l'utilisateur ai sélectionné le format de son image, on passe à l'étape suivante.

On demande alors à l'utilisateur de placer son image dans un certain dossier, et de lui donner un certain nom (4). Cette condition sur le nom de l'image permet de ne pas avoir à demander à l'utilisateur le nom de son image. L'utilisateur appuie sur la touche "espace" après avoir passé son image dans le dossier. Le programme essaie ensuite de charger l'image fournie. En cas d'échec, ceux-ci pouvant être du à un problème venant de l'image, ou bien l'impossibilité de trouver l'image si le nom ou le format de l'image fournie ne correspondent pas à ceux attendus, une page s'affiche pour informer l'utilisateur de l'échec de l'opération (5).

Il lui est suggéré sur cette page de vérifier si il a bien placé dans le bon dossier une image ayant le bon nom et le bon format. L'utilisateur est ensuite renvoyé vers la page d'instruction sur le nom de l'image. En cas de réussite, on passe à l'étape suivante.

On utilise ici une structure SDL_Rect, un rectangle, afin de redimensionner l'image pour qu'elle puisse s'afficher sans problème dans l'interface graphique, en étant centrée, redimensionnée et conservant ses proportions. On affiche ensuite l'image et on demande à l'utilisateur s'il s'agit bien de l'image qu'il a fourni (6). Cela permet à l'utilisateur de vérifier qu'il n'a pas fournit un image autre que celle qu'il souhaite utiliser. En cas de réponse négative, on revient à l'étape précédente, et en cas de validation, on passe à l'étape suivante.

C'est dans cette étape que le reste de l'OCR entre en jeux. On affiche à l'utilisateur un écran lui demandant de patienter, tandis que l'image est fournie au reste du programme, qui s'occupe du traitement (7). Une fois le traitement terminé, on change une dernière fois l'écran pour indiquer à l'utilisateur que le traitement s'est bien effectué, on lui indique dans quel fichier retrouver son texte, et enfin qu'il peut à présent quitter le programme en appuyant sur la touche "échap" (8).

Cet interface graphique a donc une apparence assez simple, mais elle permet à l'utilisateur de savoir facilement ce qu'il doit faire afin de faire traiter son image par l'OCR, ce qui était mon but ici.

## 3.8 Pré-Traitement

Le pré-traitement n'est pas essentiel pour le projet mais cela permet d obtenir un meilleur résultat.

Quentin : Je devais m'occuper de cette partie mais a cause d un manque de temps et de compétence je n'est pas pu le faire.

## 3.9 Organisation Générale du projet

En tant que chef du projet, c'est Victor qui s'est occupé de l'organisation générale du projet, des réalisations ainsi que du groupe dans sa globalité.

Victor :

Dès les premières informations concernant le projet OCR, j'ai pu organiser le groupe et répartir les différentes tâches selon les préférences de chacun. Il était alors assez évident que Vsevolod allait s'occuper du réseau de neurones et du XOR et il s'est d'ailleurs assez vite porté volontaire pour réaliser cette tâche. Martin et Quentin n'ayant pas véritablement de préférence sur les sujets restants, j'ai décidé de me placer sur la segmentation, une partie assez technique que je me sentais capable de réaliser. Martin était donc assigné à tous les algorithme de chargement de l'image ainsi que sa mise en noir et blanc et Quentin devait s'occuper des différentes fonctions de transitions entre nos parties.

À la suite de la première soutenance et donc de la première partie de travail de ce projet, les tâches ont été divisé une nouvelle fois au sein du groupe pour couvrir toutes les parties qu'ils nous restaient à effectuer. Vsevolod avait alors produit un réseau de neurones XOR fonctionnel, il a donc continué sur le réseau de neurones final qui a pour but de reconnaître des caractères. Il a été décidé que Martin allait s'occuper de l'interface graphique, une partie assez technique de l'OCR et que Quentin allait pour sa part réaliser le pré-traitement de l'image. Pour mon travail, j'ai logiquement continué et finalisé la segmentation et le découpage du texte ainsi que sa reconstruction qui sont comme dit précédemment deux parties extrêmement liée.

Ensuite j'ai pu commencer à gérer la structure générale du projet sur le dépôt git en m'informant et en codant le Makefile qui nous permet aujourd'hui de créer sans soucis notre fichier exécutable.

La gestion globale du groupe s'est avéré plus compliqué que ce a quoi je m'attendais au départ. En effet j'ai du motiver un certain nombre de fois les membres du groupe qui n'ont pas forcément produit un travail régulier pendant la première période, même s'il s'avère que nos objectifs de réalisation ont globalement été atteint pour la première soutenance.
Ce projet global que représente l'OCR n'est pas aisé à réaliser et il était nécessaire pour tous les membres du groupe de se mettre sérieusement au travail pour la seconde partie de ce projet qui est bien sur la plus importante. Pour la seconde partie de travail, il est notable que le semestre s'est encore une fois légèrement accéléré et que la travail assez important demandé par les autres cours ont souvent pris le dessus sur le travail de l'OCR en général.

La communication dans le groupe s'est révélé être un échec. Il est vrai que les quatre membres de ce groupe ne partageaient pas forcément de grandes affinités avant ce projet et que certains membres dont moi-même ne sont pas forcément très extravertis. J'ai essayé de mettre en place un contact régulier avec des moyens de messagerie instantané auxquels nous sommes familiers et que nous utilisons régulièrement mais cela n'a pas fonctionné et cela ressemblait plutôt à un monologue sans réponse.

Le dépôt git n'a également que très peu servis au vu du nombre faible d'envois du travail de chacun. J'ai donc pris l'initiative d'aller voir les membres assez régulièrement pour leur demander des nouvelles et l'avancement de leur parties.

De nombreux questionnements me viennent aujourd'hui en tant que "Chef de Groupe" pour savoir si j'ai suffisamment motivé mon équipe ou si j'ai fait défaut dans l'aspect de gestion globale du projet. Je pense avoir essayé de faire le maximum de mon côté pour souder le groupe et pour produire un projet de la plus grande qualité possible mais il est difficile de me faire un avis sur ce sujet étant donné le peu de retour que j'ai au niveau de mon groupe.

Par comparaison, je ne retrouve pas cet esprit qui animait mon groupe de projet durant le second semestre de la première année, cet esprit presque combatif plein d'envie et de motivation qui nous avait alors, malgré les difficultés et la charge de travail, permis de produire un projet qui se voulait le plus professionnel possible.

Pour conclure ici, il est vrai que quelque soit l'endroit où nous travaillerons tous plus tard dans n'importe quelle discipline il faudra s'adapter à ses collaborateurs et à tous ceux qui travailleront avec nous et donc que ce projet nous apporte un peu plus d'expérience et de connaissance sur le travail en groupe en général ainsi que sa gestion.

# 4 Réussites Techniques

Voici donc ici les différentes réussites technique de tous les membres du groupe. En effet ce projet n'est pas si aisé que ça et il est important de prendre du recul et de dégager du positif dans cet OCR.

Victor :

Au niveau technique pour commencer, la majeure partie de mon travail qui est la segmentation et le découpage du texte ainsi que la reconstruction de ce dernier est une réussite globale. Ma partie, bien qu'elle puisse être presque indéfiniment amélioré, fonctionne correctement et permet dans de très nombreux cas de découper correctement les caractères sur une image, de les envoyer aux réseaux de neurones pour la reconnaissance, puis de reconstruire le plus fidèlement possible le texte de départ pour l'écrire dans un fichier texte que l'utilisateur pourra facilement retrouver et manipuler.

Ensuite au niveau de l'apprentissage et de l'utilisation du langage de programmation C, je suis assez content des compétences que j'ai pu acquérir au cours de ces trois mois assez intense. Il est clair qu'il va falloir retravailler certaines notions un peu plus techniques et fondamentales pour pouvoir bénéficier du potentiel maximum et des avantages de ce langage.

Un peu plus précisément, l'apprentissage et l'approfondissement de l'utilisation de la SDL, module qui permet de manipuler des images, était très intéressant. Cela liée à ma partie de segmentation et de découpage du texte, une partie très tourné vers l'algorithmie pur, m'a permis de prendre du plaisir à faire ma partie malgré les difficultés de ce projet.

Quentin :

Malheureusement mes fonction ne marche pas mais ce projet m'as appris a m amélioré dans le langage C, et j ai pu découvrir le fonctionnement de sdl et un petit peu le sdl2

Martin :

Je suis personnellement satisfait du travail que j'ai effectué sur l'interface graphique. En effet, bien que je n'ai pas pu ajouter certaines fonctionnalités que j'aurais souhaité pouvoir faire fonctionner, l'interface graphique est opérationnelle. Je trouve son fonctionnement assez intuitif, ce qui permet de ne pas perdre l'utilisateur dans une interface trop complexe.

Cependant, elle est assez développée, et n'est pas un simple fenêtre affichant "Appuyez sur une touche ... traitement terminé". En effet, l'utilisateur a un certain degré d'interaction avec l'interface, ce qui est plus confortable.

Au niveau du passage en noir et blanc de l'image, je suis content d'avoir pu un minimum améliorer cette partie, qui était très sous-développée lors de la première soutenance, en ajoutant le système permettant d'obtenir la couleur moyenne de l'image, ce qui permet de séparer plus aisément le texte du fond.

Vsevolod :

Mon principal succès est la création d'une structure matricielle adéquate, fonctionnelle et indexée, avec plusieurs fonctions d'analyse et d'interaction avec les matrices. Les matrices étant la base de tout bon réseau, je suis plutôt fier du résultat que j'ai obtenu.
De plus, je comprends la fonction et le but de chaque détail situé à l'intérieur de mon réseau, même s'il y a un bug que je ne parviens pas à trouver au moment de la rédaction de ce rapport, ce qui freine plutôt mon enthousiasme.

Je suis cependant assez fier de vous présenter le système save/load, qui fait son travail avec vigueur et style. Bien que mon réseau ne parvienne pas tout à fait à m'obtenir un résultat souhaité, vu que je n'ai pas de choix particulier en la matière, j'ai accepté que, pour les besoins du projet, je n'ai pas réussi à terminer ma partie dans le temps, mais je suis content de savoir que je comprends comment fonctionne un réseau neuronal, et serait capable de corriger l'erreur si j'avais une légère extension.

# 5 Complications Techniques

Bien sur, le projet n'est pas tout noir ni tout blanc, et même des parties réussies ont été implémenté avec de nombreuses complications et de nombreux soucis. Les membres du groupes présentent donc ici ce qui techniquement a été le plus difficile au cours de cette période de travail.

Martin :

Ma principale déception fut de ne pas pouvoir implémenter la rotation d'image. En effet, j'avais comme idée d'ajouter à l'interface graphique un outil permettant d'effectuer à l'image une rotation. celle-ci aurait permis à l'utilisateur de tourner l'image comme il le souhaite avant de l'envoyer au traitement. cela aurait pu permettre à l'utilisateur de corriger l'orientation de l'image si, par exemple, le document original n'aurait pas été placé droit dans un scanner, ou bien si l'image était totalement à l'envers et nécessitant une rotation à 90°.

Cependant, bien que la partie interface de cette rotation était fonctionnelle, la seule chose qui effectuait la rotation était le renderer affichant l'image, et pas l'image elle même. J'avais donc prévu d'ajouter la rotation de l'image lors du passage en noir et blanc. cependant, cela étant assez complexe, j'ai décidé d'abandonner ce projet. En effet, je pense que cela aurait été réalisable, mais le temps manquait, et ce n'était pas une priorité, j'ai donc préféré me concentrer sur des taches plus importantes.

Quentin :

Malheureusement mes fonction ne marche pas, il y a pas mal de warning. De plus j ai eu beaucoup de mal avec le langage C en plus de mes difficulté de base en programmation. Ensuite j ai commencé a coder en sdl puis Martin m as dis qu'il utilisait le sdl2, car la documentation était plus fourni et que la gestion des fenêtres étaient un peu plus simple.J ai donc commencé a changé le sdl en sdl2 mais cela ne marchait pas du tout, je suis donc repasser sur du sdl.

Victor :

Il est vrai que même si ma partie est plutôt un succès dans sa globalité, ce projet m'a confronté à de nombreuses difficultés au niveau technique et programmation.

Le premier élément qui m'a pris un certain temps et qui n'a pas pu être implémenté est un algorithme viable pour le calcul de la valeur de l'écart des pixels noirs dans mon algorithme de RLSA. Cette composante m'avait été justement reproché lors de la première soutenance et j'avais alors pris la décision de m'occuper de cette partie. Malheureusement malgré quelques tentatives et idées, comme par exemple utiliser la hauteur et la largeur de l'image ou encore d'exploiter la taille du premier caractère rencontré, je n'ai pas pu trouver un algorithme qui me présentais des résultats assez fiable sur un assez grand nombre d'exemples, en tout cas pas plus fiable que des valeurs génériques qui me permettent un RLSA correcte. Malgré cette échec technique, les conséquences ne sont que minimes pour l'OCR, le plus grand risque étant "seulement" de perdre un saut de ligne entre les paragraphes et au pire des espaces entre certains mots.

Le second point qui m'a valu des complications durant cette période de travail est le cas où les caractères sont collés entre eux et qu'ils n'y a même pas un seul pixel blanc qui les sépare. En effet, ce cas est assez dur à gérer dans mon implémentation. L'algorithme que j'ai mis en place pour résoudre cette difficultés se basait sur le nombre de pixels noirs rencontrés lors du parcours vertical des pixels sur une ligne donnée, mais il a rapidement présenté un défaut majeurs qui était de couper certaines lettres comme le "u", le "j" ou encore le "h"aux endroits où elles possèdent peu de pixels noirs dans leur axe vertical. Je n'ai malheureusement pas pu corriger cela par manque de temps et par priorité d'optimisation des parties déjà fonctionnelles pour pouvoir obtenir un premier OCR simple mais totalement viable.

Le dernier regret que je peux exprimer est celui du traitement des images dans ma segmentation. Je m'étais fixé cette problématique comme tout premier "bonus" lorsque tout mon découpage était fonctionnel mais je n'ai pas pu m'y pencher de façon approfondi pendant ce projet.

Je peux ici conclure en notant qu'il est toujours possible d'aller chercher plus loin et plus d'optimisation dans un projet comme celui-ci, ce qui lui permet en plus de générer de l'intérêt d'être une source d'apprentissage et de développement des compétences très importante.

Vsevolod :

Le plus grand problème auquel j'ai dû faire face lors de l'exécution technique du projet, était le fait que j'ai dû adapter la bibliothèque Numpy en C afin de l'utiliser pour la construction du réseau. N'ayant pas travaillé sur un support stable, j'ai été confronté au fait qu'à un moment donné, j'ai perdu la totalité de mon travail sur lequel j'avais passé une soirée entière, pour avoir à le refaire le lendemain soir. Ma confusion concernant l'utilisation et la fonction de l'allocation de mémoire dynamique n'a rien fait non plus pour améliorer ma compréhension.

Mais, je crois que le facteur le plus invalidant dans mon travail, c'est que je ne peux tout simplement pas sembler simplement demander l'aide de quelqu'un, que ce soit en ligne ou hors ligne. En fait, je me suis retrouvé à crier devant mon écran plus de fois que je n'ose l'admettre, et une simple seconde paire d'yeux pendant cette période m'aurait énormément aidé, le fait que cela me pique un peu l'estime de moi-même maintenant que la présentation finale de notre travail approche.

# 6 Bilan Personnel

Martin :

Maintenant que ce projet est terminé, mon ressentiment de celui-ci est assez complexe. En effet, j'ai trouvé que ce projet était très intéressant, et je suis content du travail que j'ai fournit et des connaissances que j'ai acquis.

Cependant, il y a selon moi un point dans ce projet qui me pose problème, et qui ressort encore plus quand je compare ce projet à celui de l'année dernière, et c'est le manque de communication dans notre groupe. En effet, il est impossible d'ignorer que notre communication en temps que groupe était plus que limitée. Je pense que cela était en partie du à notre manque de contact entre nous. En effet, nous ne nous voyons que très rarement, nous n'avons pas eu réellement de réunions pour travailler ensemble et évaluer l'avancement du projet...

Nous nous répartissions simplement les taches et travaillons chacun de notre coté. Cela peut être observé sur le tableau de répartition des taches, où chacun a sa partie, personne ne partage de parties et aucun suppléant n'existe nulle part. Il y a pourtant eu des efforts pour améliorer la communication du groupe, notamment de la part de notre chef de groupe, Victor, mais sans réel effet. Je ne penses cependant absolument pas que ces échecs étaient de sa faute, je pense moi-même être incapable de faire ce qu'il a fait. Le problème venait plutôt, selon moi, du groupe dans son ensemble, et de nos relations les uns envers les autres.

Ce manque de temps passé ensemble dès le début a fait que nos relations entre nous n'ont jamais réellement évolués, et nous sommes restés uniquement des membres de groupe. Le dialogue entre nous était très limité et j'ai quelque fois discuté du projet avec Victor dans les couloirs entre deux cours, ou ai essayé de parler avec Quentin et Vsevolod de leur avancement du projet, mais nous n'avons jamais réellement agis en temps que groupe uni dans un projet commun, et plutôt comme des individus travaillant séparément sur des parties d'un grand projet.

En conclusion, je pense que la communication était le gros point faible de notre groupe. Je sais cependant que je serais sans doute amené à travailler dans le futur sur des projets avec des conditions similaires, et je compte apprendre de ces erreurs afin de ne pas les laisser se répéter.

Quentin :

Malheureusement mes fonction ne marche pas mais ce projet m'as appris a travaillé avec des gens avec lesquels je ne suis pas habitué mais malgré cela le projet s est plutôt bien déroulé et la communication était plutôt bonne entre tous les membres du groupe. Ce projet ce rapproche beaucoup plus de ce que l on pourrait retrouvé en entreprise, par rapport au projet de première année.
De plus le projet m as permis de m amélioré dans un nouveau langage dans lequel j ai beaucoup de difficulté, le C.

Vsevolod :

Il n'y a pas grand-chose à dire sur la communication du groupe, qui était, pour être exact, assez absente de l'ensemble du projet, et je ne suis certainement pas exclu de ce problème. Être introverti a ses inconvénients, après tout. Pourtant, je dois quand même noter que la plus grande faiblesse que j'ai démontrée au cours de ce projet a été le fait que j'ai pris la voie facile du poing, ce qui m'a empêché de poursuivre mes études afin de devenir tout simplement trop difficile à gérer seul. Et, encore une fois, je n'ai demandé l'aide de personne alors que j'aurais vraiment dû.

Victor :

Je crois que mes coéquipiers ont assez résumé l'ensemble du projet dans cette partie. Au niveau de mon ressenti personnel, je pense en avoir déjà expliqué une bonne partie au niveau de la gestion globale du groupe. Néanmoins je peux encore ici exprimer ma déception globale au niveau de l'OCR. En effet, avec le recul que je peux avoir aujourd'hui alors que le projet a sa forme final, je ne peux que noter qu'à aucun moment dans ce projet je me suis sentie vraiment porté par celui-ci. C'est une sensation assez difficile à expliquer mais qui est bien présente et qui me fait penser que je suis passé à coté de quelque chose.

Les problèmes de communications global, le manque de motivation ou de travail ne nous ont pas permis de rendre un projet rien que fonctionnel et lié. Cela, sûrement due au fait que je suis très perfectionniste et que j'aime rendre dans n'importe quel matière un travail qui est fini et qui est perfectionné le plus possible.

Je suis néanmoins assez content d'avoir pu faire marcher mon travail et mes différents algorithmes, et je suis assez satisfait de mon travail. Ce projet a été un expérience intéressante et cela est également due aux nombreuses complications rencontrés. Cela me rappelle ce pourquoi j'ai choisi EPITA, me rapprocher le plus vite possible d'exemples et de projets concrets et réalistes.

# 7 Coût de production

Un projet c'est bien, mais sans financement, c'est rien. Voici le coût de production de notre projet. Le matériel, les logiciels, tout est pris en compte, mais nous avons un budget très serré ! C'est pourquoi nous serons certainement un des groupes les plus économes de cette promotion !

| **Achats** | **Coût** |
|---|---|
| Année à l'EPITA x 8 (en fait un peu plus) | 64000 € |
| Ordinateur faisant tourner EMACS x 4 | 4000 € |
| Connection Internet 3 mois x4 | 800 € |
| Cargaisons de Corn Flakes | 200 € |
| Silos de bananes | 2000 € |
| DVD, Jaquette, Manuel | 20 € |
| ~~Pot de vin~~ Cadeau pour Mme Derrode | 666 € |
| Frais de soins pour le manque de sommeil de Victor | 9999 € |
| Fête la veille de la soutenance | $10^{42}$ € |
| Sac à dos en peau d'ours Ultra résitant pour porter Boffelli | 4242 € |
| **Total** | $\infty$ |

**Coût de production (C'est plus cher qu'au S2 non ?)**

# 8 Conclusion

En conclusion, nous sommes tous satisfais du travail que nous avons effectués individuellement, et avons apprécié travailler sur ce projet.

Cet OCR nous a appris de nombreuses choses dans différents domaines que nous n'aurions sans doute pas abordé autrement, et ces connaissances nous seront sans doute utiles dans le futur.

Notre principal regret sera cependant de ne pas avoir réussi à réunir nos différentes parties afin d'assembler un projet complet et fonctionnel. En effet, le manque de communication dans notre groupe nous a empêchés de correctement connecter nos parties entre elles, et même si le fonctionnement général du projet est majoritairement opérationnel, nous n'avons pas eu le temps de les connecter, et elles restent des pièces d'un puzzle non assemblé.

Nous avons tous cependant appris beaucoup de choses durant ce projet, et gardons un souvenir positif de cet OCR, malgré les quelques regrets.

*"C'est ici que se ferme le rapport ainsi que l'OCR"*

# 9 Bibliographie

https://fr.wikipedia.org/wiki/Seuillage_d'image

https://fr.wikipedia.org/wiki/Méthode_d'Otsu

https://fr.wikipedia.org/wiki/Binarisation

https://medium.com/@susmithreddyvedere/segmentation-in-ocr-10de176cf373

https://www.mathworks.com/help/vision/examples/recognize-text-using-optical-character-recognition-ocr.html

http://how-ocr-works.com/OCR/line-segmentation.html

https://crblpocr.blogspot.com/2007/06/run-length-smoothing-algorithm-rlsa.html

https://www.tutorialspoint.com/dip/grayscale$_t o_r gb_c onversion.htm$

https://wiki.libsdl.org/FrontPage

# 10 Annexes

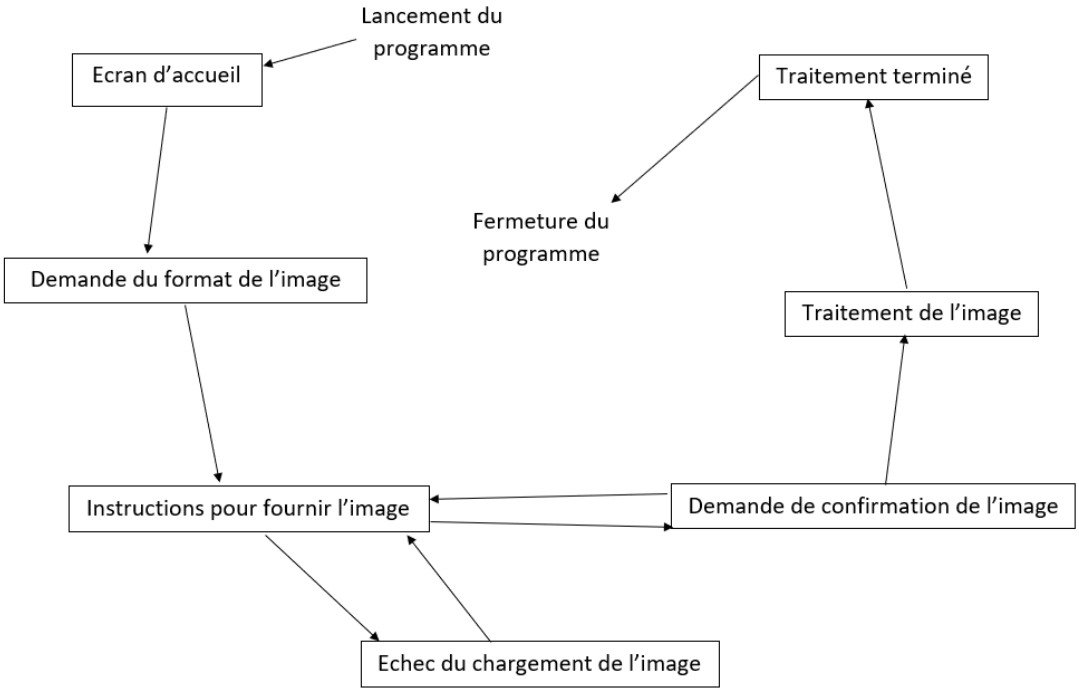


FIGURE 1 – Plan des étapes l'interface graphique


FIGURE 2 – Écran d'accueil

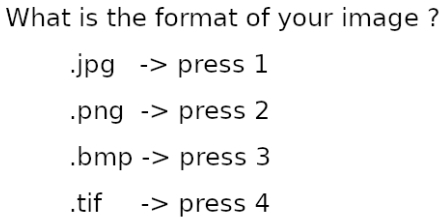


FIGURE 3 – Demande du format de l'image

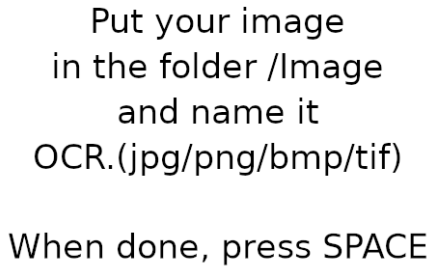


FIGURE 4 – Instructions pour fournir l'image


FIGURE 5 – Échec du chargement de l'image

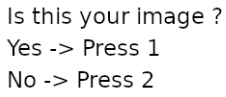


FIGURE 6 – Demande de comfirmation de l'image

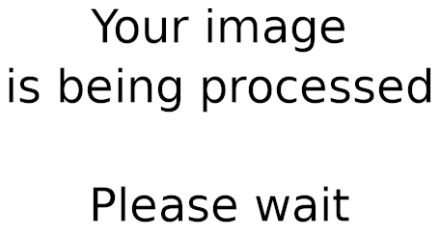


FIGURE 7 – Traitement de l'image

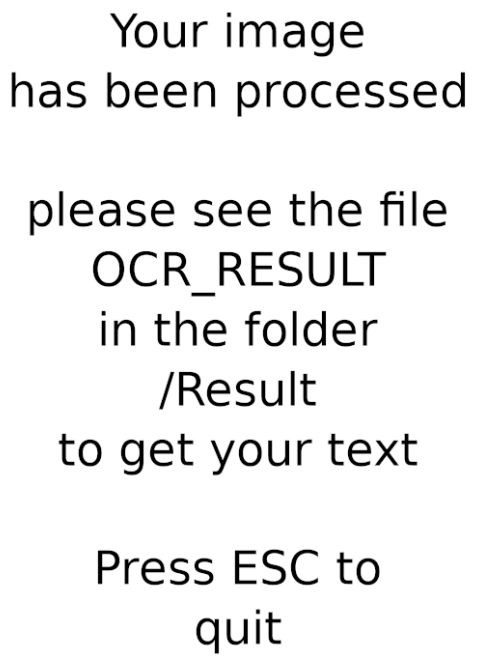


FIGURE 8 – Traitement terminé

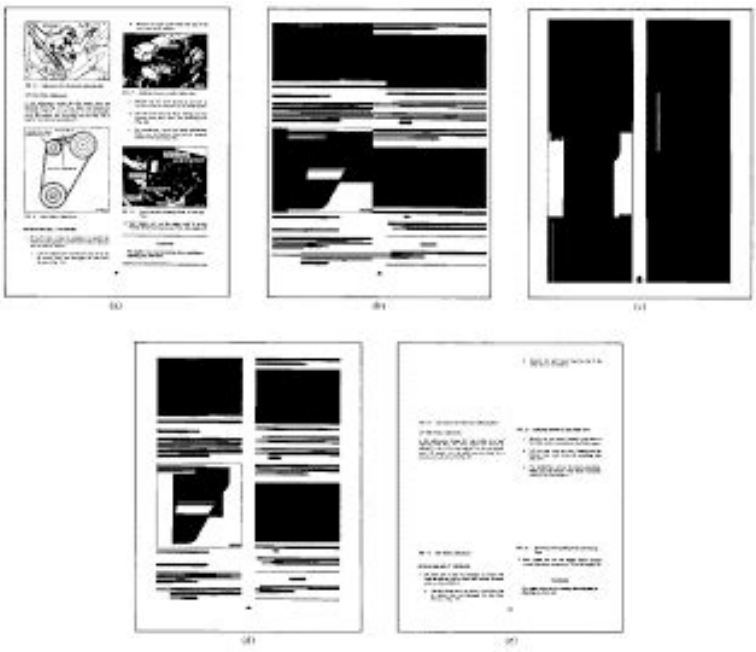

FIGURE 9 – Run Length Smoothing Algorithm (RLSA)

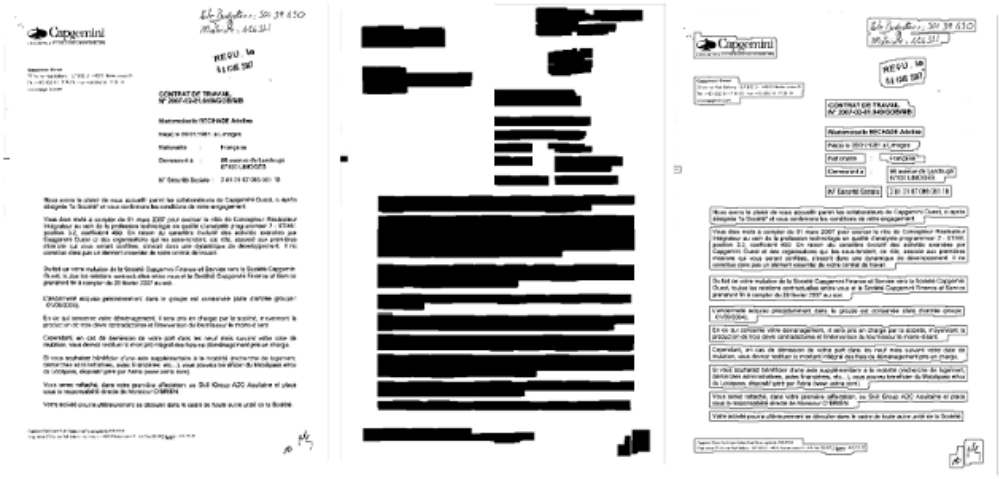

FIGURE 10 – Découpage de l'image

Algorithmique

Hachage : Principe et méthodes de bases

1 Principe du hachage

On utilise les méthodes de hachage lorsque l'on doit classer les éléments d'une collection appartenant à un univers très grand. La taille de ce dernier nous empêche de le représenter en réservant une place mémoire par clé possible. On utilise alors une fonction de hachage h qui associe à chaque clé un entier compris entre 0 et m – 1 (ou 1 et m), où m est choisi en fonction de la taille prévue de la collection.

Pour toute clé x de la collection, h(x) (valeur de hachage primaire) donne l'indice de x dans le tableau de hachage. Cela nous permet de le rechercher, l'ajouter ou le supprimer. Le choix de la fonction de hachage est fondamental, celle-ci doit être :

uniforme : c'est à dire que tous les éléments sont répartis le plus uniformément possible dans le tableau, facile et rapide à calculer : le but étant de ne pas affaiblir la méthode par un calcul long et fastidieux, déterministe : renvoyer toujours le même résultat.

Si la fonction h est injective, il n'y a aucun problème : chaque élément aura sa place unique dans le tableau. Si ce n'est pas le cas, plusieurs élément peuvent avoir la même valeur de hachage, on parle alors de collision. Les méthodes de résolutions des collisions (directes et indirectes) seront vues dans des cours spécifiques.

FIGURE 11 – Algorithme du XY-Cut

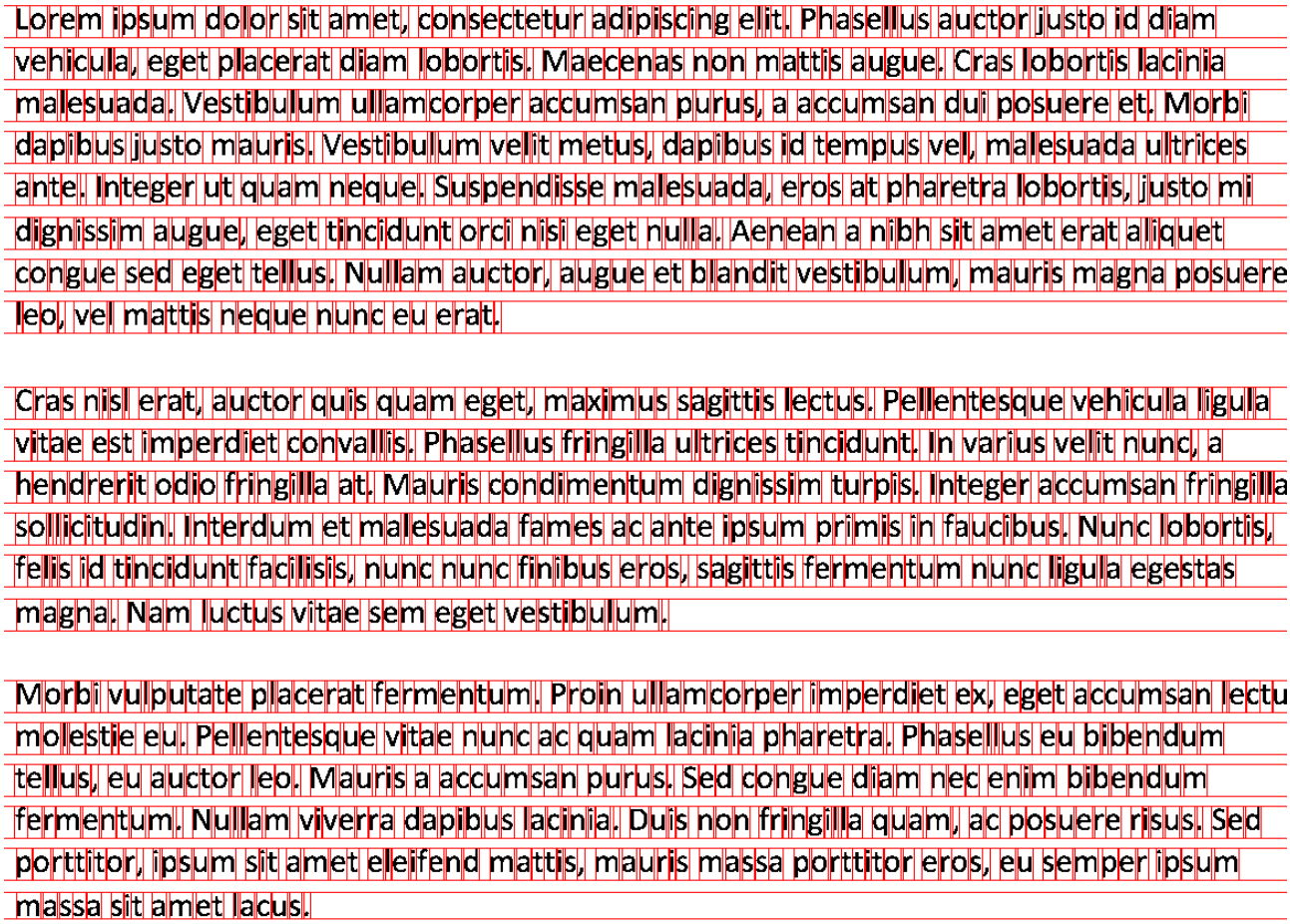


FIGURE 12 – Algorithme du XY-Cut(2)

```
00000 00000 00000 000 00000 00000000000 0000000000 00000 000000000 000000 00000 00 0000
000000000 0000 00000000 0000 000000000 00000000 000 000000 000000 0000 00000000 0000000
0000000000 0000000000 00000000000 00000000 000000 0 00000000 000 0000000 000 00000
0000000 00000 0000000 0000000000 00000 000000 0000000 00 000000 0000 000000000 00000000
00000 0000000 00 0000 000000 00000000000 0000000000 0000 00 00000000 000000000 00000 00
000000000 000000 0000 000000000 0000 0000 0000 000000 000000 0 0000 000 0000 0000 0000000
000000 000 0000 0000000 000000 0000000 00000 00 0000000 00000000000 000000 00000 0000000
0000 000 000000 00000 0000 00 00000

0000 0000 00000 000000 0000 0000 00000 0000000 00000000 0000000 000000000000 00000000 000000
00000 000 000000000 0000000000 000000000 000000000 00000000 0000000000 00 000000 00000 00000 0
000000000 0000 000000000 000 000000 00000000000 000000000 0000000 0000000 00000000 000000000
0000000000000 00000000 00 000000000 00000 00 0000 00000 000000 00 000000000 0000 000000000
00000 00 000000000 0000000000 0000 0000 0000000 00000 00000000 000000000 0000 000000 0000000
000000 000 000000 00000 000 0000 00000000000

00000 000000000 00000000 0000000000 00000 00000000000 000000000 000 0000 00000000 000000
00000000 000 000000000000 00000 0000 00 0000 0000000 000000000 000000000 00 00000000
0000000 00 000000 0000 000000 0 00000000 000000 000 000000 0000 000 0000 00000000
0000000000 000000 0000000 0000000 00000000 0000 000 000000000 00000 00 0000000 000000 000
0000000000 00000 000 0000 00000000 0000000 000000 00000 000000000 00000 00 000000 00000
00000 000 0000 000000
```

FIGURE 13 – Exemple de texte de sortie