Rapport de Projet
Projet-2028-S3
Word Spotters
OCR Word Search Solver

Misandeau Clément
Rital Marwan
Baracand Romain
Durand-Marliac Julien

Table des matières

1 Introduction

A Présentation du Projet (OCR Word Search Solver)

L'objectif de ce projet est de réaliser un logiciel de type OCR (Optical Character Recognition) qui résout une grille de mots cachés.

Dans sa version définitive, notre application proposera une interface graphique permettant de charger une image, de la visualiser, de corriger certains de ses défauts, et enfin d'afficher la grille complètement remplie et résolue. Ce projet est réalisé dans le cadre des études en cycle préparatoire de deuxième année de l'école EPITA.
Nous nous sommes rassemblé en un groupe de 4 membres, qui se nomme "WordSpotters".

B Qu'est-ce que la reconnaissance optique de caractères (OCR) ?

La reconnaissance optique de caractères (OCR) est le processus permettant de convertir une image de texte en format de texte lisible par une machine.

Pour illustrer ce propos, si l'on scanne un formulaire ou un quelconque texte imprimé, un ordinateur enregistre la numérisation en tant que fichier image. Cependant on ne peut pas utiliser par exemple un éditeur de texte pour modifier le fichier, y faire une recherche ou en compter les mots. Par contre, on peut utiliser l'OCR pour convertir l'image en document texte, dont le contenu sera stocké en tant que données textuelles. Ce contenu pourra être utilisé et modifié par de nombreux logiciels plus aisément.
Elle est donc couramment utilisée pour reconnaître le texte dans les documents et images numérisés

2 Présentation des Membres

A Marwan Rital

Je suis étudiant en deuxième année à l'école *EPITA* de *Villejuif*. Originaire de Dijon, je suis passionné par différents langages de programmation tels que Python et Javascript avec NodeJS. Par la suite, je souhaiterais m'orienter vers le domaine de l'intelligence artificielle, qui me passionne tout particulièrement.

Depuis l'an dernier, avec le développement de notre projet "Memento Mori", j'ai appris de nouvelles compétences techniques et méthodologiques essentielles. La gestion de la structure du code, son optimisation ou encore les problématiques liées aux fonctionnalités du projet. En parallèle, j'ai appris à m'organiser davantage et créer un environnement propice au développement afin de réaliser les objectifs et établir un planning rigoureux des engagements à respecter. Cette expérience m'a également aidé à travailler en équipe grâce à des échanges réguliers concernant les avancées et les problèmes associés.

En utilisant ces compétences acquises lors de mon précédent projet, j'aborde "WordSpotters" avec une meilleure compréhension des attentes et de la méthode à utiliser pour assurer une finalité optimale. L'apprentissage des bases de l'OCR en utilisant des bibliothèques comme SDL est enrichissant, permettant de mieux comprendre les limites et les utilisations plus globales de cette dernière.

La mise en œuvre de ce projet en C est parfaitement adaptée. Effectivement, ayant comme objectif de renforcer et approfondir mes connaissances dans ce langage, "WordSpotters" représente une opportunité idéale.

B Julien Durand-Marliac

Je suis actuellement étudiant en deuxième année du cycle préparatoire de l'EPITA. J'ai rejoint cette école car je suis intéressé par l'informatique et les nouvelles technologies.

« Word Spotters » est pour moi une très bonne occasion de découvrir et d'apprendre à maîtriser de nouveaux domaines de l'informatique. En effet lors du précédent projet auquel j'ai participé (Necro.io), il a été utilisé un algorithme de masse pour Intelligence Artificielle, ce qui n'est pas ce à quoi je m'attendais et qui ne m'a pas laissé explorer le monde des IA autant que je l'aurais souhaité. Dans ce projet en revanche, il m'est offert l'opportunité de découvrir le deep-learning et d'apprendre à mettre en pratique mes connaissances dans un nouveau langage de programmation, le langage C.

Je suis, en conclusion, très motivé pour ce projet, tant par la nouveauté que par l'expérience qu'il m'apporte.

C Romain Baracand

Depuis mon enfance, j'ai toujours eu un vif intérêt pour l'univers du jeu vidéo, puis, en grandissant, c'est celui pour l'informatique qui n'a cessé de se développer. C'est au cours du confinement que j'ai touché pour la première fois à la programmation, et depuis il m'a toujours paru évident que la voie qui me conviendrait le mieux serait celle de l'informatique.

C'est donc pour faire de ma passion mon avenir que j'ai choisi de prendre la spécialité NSI au lycée et ainsi intégrer l'EPITA, où j'effectue actuellement ma deuxième année.

La pratique est selon moi le meilleur moyen d'apprendre, et le projet de l'année dernière m'ayant permis d'apprendre plusieurs compétences, je suis confiant de pouvoir tirer de ce projet beaucoup d'expérience, notamment sur le fonctionnement des logiciels de reconnaissance optique et sur l'intelligence artificielle.

D Clément Misandeau

Je suis étudiant à l'école *EPITA* en 2ère année de cycle préparatoire. Pour me présenter brièvement, je viens d'une terminale générale avec les spécialités mathématique et informatique. Avec ce bagage en termes de connaissances et de hobbies, j'ai décidé de m'orienter dans des études longues en informatique.

Depuis le dernier projet, que j'ai réalisé durant ma première année, j'ai gagné de nouvelles compétences concernant la gestion d'un projet sur plusieurs mois. "MementoMori" reste pour moi un projet qui m'a permis de me surpasser et qui m'a donné goût au travail sur le long terme.

Personnellement, en entrant dans ce nouveau projet je découvre complètement les logiciels de reconnaissance optique simple de caractères. Cependant, il est certain que ce sujet me motivera car le traitement d'image est un domaine qui m'intéresse depuis longtemps mais avec lequel je n'ai fait qu'effleurer. Ce sera l'occasion pour moi de progresser et d'apprendre dans ce domaine.
Ce projet aura donc pour but personnel, de comprendre ce qu'est un OCR mais aussi d'améliorer mes connaissances en C et mes capacités concernant la conduite d'un projet sur une plus petite durée.

Ce deuxième projet est donc une nouvelle occasion de progresser et de partager avec les autres étudiants de mon groupe. Ce projet est à prendre comme une expérience enrichissante avant tout.

3 Répartition des tâches

A Répartition des taches

Dans le tableau ci-dessous, vous pouvez voir la répartition des taches en fonction de chaque membre. Dans cette liste de tâches, chacun a été affecté en fonction de ses préférences et de ses choix. Cependant dû au caractère complexe du projet, car il touche à de nouvelles notions, il est important d'indiquer que chacun est libre d'aider sur d'autres parties du projet.

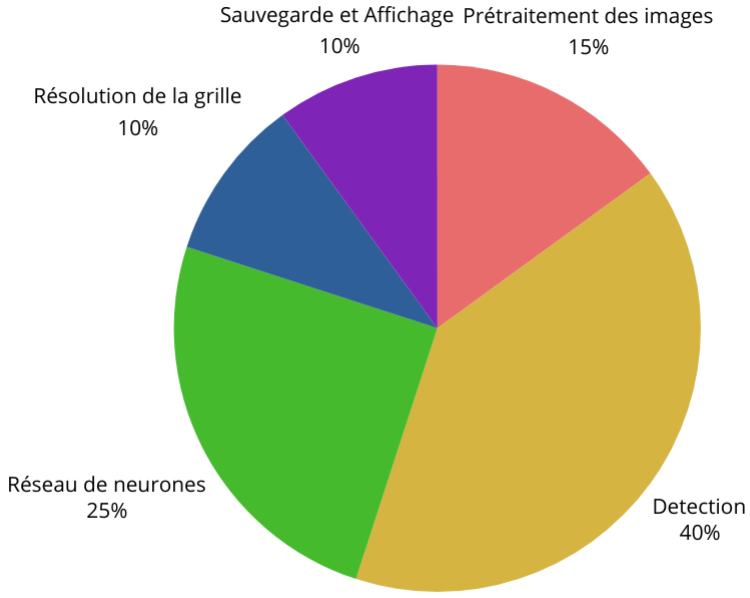
Tâches	**Clément**	**Marwan**	**Julien**	**Romain**
Chargement d'une image	⊕			
Suppression des couleurs	⊕			
Prétraitement	⊕		⊖	
Détection de la position grille		⊕		⊕
Détection de la position de la liste de mots	⊖	⊖	⊖	⊖
Détection des cases de la grille		⊕		⊕
Récupération sous la forme d'image		⊕		
Reconnaissance de caractères à partir des images	⊖		⊕	
Reconstruction de la grille		⊖	⊖	⊖
Reconstruction de la liste	⊖	⊖	⊖	
Résolution de la grille				⊕
Affichage de la grille résolue	⊖			⊕
Sauvegarde de la grille résolue	⊖			⊕

Légende :
Tache Principale = ⊕ ; Tache Secondaire = ⊖

Durant le développement du projet, nous mettrons en ligne un graphique de tache qui sera mis à jour tout au long de celui-ci. Chaque membre pourra voir les différents avancements des autres parties et avoir une meilleur conception sur l'avancée globale.
Ce graphique d'avancement se divisera en deux parties distinctes, qui prendront en compte les deux jalons des soutenances à venir.

B Répartition du temps

Le diagramme ci-dessous illustre la répartition du temps attribuée à chacune des grandes étapes du projet. C'est une conclusion graphique résultant de notre partage des tâches, nous permettant d'identifier au premier coup d'œil les travaux chronophages entre les tâches.



Tâches	Répartition du temps en %
Détection	40%
Réseau de Neurones	25%
Résolution de la grille	10%
Sauvegarde et Affichage	10%
Prétraitement	15%s

4 Développement Technique

A Prétraitement des images

Présentation du chargement des Images

La mécanique de chargement d'une image se trouve dans le fichier MAIN.C. Pour la charger nous avons utilisé la bibliothèque de SDL2, elle nous permet à la fois de charger les fichiers mais aussi de les afficher à l'utilisateur.

Afin de pouvoir utiliser l'image fournit par l'utilisateur, nous utilisons une variable SDL_Surface pour stocker et modifier l'image. Pour acceder aux differentes propriétés de l'image comme ses pixels, nous utilisons la fonction SDL_ConvertSurfaceFormat pour modifier le format de la surface.

```
...
SDL_Surface * image = IMG_Load(argv[1]);
SDL_Surface * originalImage = image;
    image = SDL_ConvertSurfaceFormat(image, SDL_PIXELFORMAT, 0);
    SDL_FreeSurface(originalImage);
...
```

Présentation de la sauvegarde

La mécanique de sauvegarde d'une image se trouve dans le fichier SAVE.C. Pour sauvegarder l'image traitée on fait une capture grace à SDL_CreateRGBSurface et SDL_RenderReadPixels pour sauvegarder la surface.

Présentation de la fonction Grayscale et BlacktoWhite

Les deux mécaniques se trouvent dans le fichier GRAYSCALE.C. Elles comportent une même structure où l'on parcours tout les pixels d'une surface.
Seulement là où Grayscale change les pixels en fonction de la formule : "0.21 * r + 0.71 * g + 0.07 * b" où r,g et b sont des entiers. ToBlackWhite les modifie certes en fonction de même la formule mais avec en sortie seulement les couleurs : blanche et noire.

```
    ...
    // Use of the formula for the grayscale function
    Uint8 v = 0.212671f * r + 0.715160f * g + 0.072169f * b;
    // Opposite method for the extraction of rgb
    pixel = (0xFF << 24) | (v << 16) | (v << 8) | v;
    pixels[y * image->w + x] = pixel;
    ...
```

Présentation de la fonction de rotation

La mécanique de rotation n'est pas complètement implémentée, en effet elle ne comporte que quelques angles. Les angles retenus par la fonction sont 90°,180° et 270°. Elles correspondent à des rotations "simples" car tout les pixels d'une image A restent dans l'image B. Il n'y a pas de perte ou d'ajout.
Afin de tester différents moyens de modifier des images, nous avons utilisé une nouvelle méthode qui est indépendante de SDL. Pour cela on utilise la bibliothèque de stdio.h avec les fonctions

"fopen","fwrite" et "fclose".

Cependant, la rotation n'est pas fonctionnelle.

Présentation de la fonction de luminosité

La mécanique de changement de luminosité d'une image permet d'éviter la perte d'information lors de son traitement. Ainsi elle nous évite de perdre certaines lettres ou grilles.

Son fonctionnement est une combinaison de deux fonctions. L'une permet d'extraire les informations et de les modifier. L'autre effectue les calculs pour le changement du pixel.

```
Uint8 f(Uint8 c, int n)
...
void Brightness(SDL_Surface * image, int n)
...
```

Dans la fonction "Brightness", on extrait chaque donnée des pixels (composition RGB) pour les passer dans l'autre fonction. On les remet en place avec cette ligne :

```
pixel = (0xFF << 24) | (r << 16) | (g << 8) | b;
pixels[y * image->w + x] = pixel;
```

B Détection

Après avoir effectué les actions de prétraitement telles que la rotation et la mise en noir et blanc, il faut maintenant détecter la position de la grille et de la liste de mots dans l'image, ainsi que la position de chaque lettre de la grille et de la liste.
C'est à cela que sert le fichier DETECTION.C, qui peut être utilisé après compilation via make par la commande :

```
./detection <nom de fichier>
```

Où le fichier donné correspond à une image ayant subi un prétraitement (c'est-à-dire une image ayant subi une rotation pour être "droite" ainsi qu'une suppression des couleurs, car nous n'avons besoin que d'images en noir et blanc pour effectuer cette action).

Après avoir lancé le programme, l'image va s'ouvrir dans une nouvelle fenêtre, et il sera possible de lancer la détection des lettres. Au terme de la recherche, toutes les lettres apparaîtront entourées par des rectangles oranges sur l'image ouverte afin de montrer les résultats du programme.
Cette partie à été la plus compliquée à implémenter jusque là pour notre groupe, et ne fonctionne que partiellement sur les images plus complexes.

La méthode que nous avons choisi d'adopter et de tout d'abord parcourir l'image pixel par pixel.

```
void detectionLoop(SDL_Surface* image)
{
    Uint32 * pixels = (Uint32 *)image->pixels;
```

```
int listSize = 0;
for (int y = 0; y < image->h; y++)
    for (int x = 0; x < image->w ; x++)
        ...
```

Pour chaque pixel sombre rencontré, on crée une "zone", qui est ensuite étendue à chaque pixel sombre adjacent, pour qu'à terme cette zone contienne l'entièreté du caractère. Ensuite, les coordonnées de cette zone sont sauvegardées dans une liste dynamique, après vérification qu'elle ne soit pas un doublon d'une zone existante.

```
int isInList(int **list, int *listSize, int x1,
    int y1, int x2, int y2, int size)
    ...
```

Par manque de temps et à cause de difficultés techniques, il n'a pas été possible de programmer l'étape finale, qui correspond au tri entre les éléments de la grille, de la liste, et du "bruit", c'est-à-dire tout ce qui n'est ni dans la grille ni dans la liste de mots.
Cependant, nous avons réfléchi à une solution pour ce problème, qui consisterait à reconnaître la grille par le regroupement de plusieurs "zone", ayant un espacement constant et formant un rectangle, et à reconnaître la liste par un regroupement de plusieurs "zones" ayant un espacement constant mais ne formant pas forcément un rectangle car les mots de la liste ont généralement des longueurs différentes.

De plus, nous avons tenté auparavant d'implémenter un système permettant de distinguer les 3 catégories (lettres de la grille, lettres de la liste de mots, et "bruit") en utilisant les tailles de chaque zone ainsi que la taille moyenne d'une zone, mais les résultats n'étant pas concluants, nous avons abandonné cette idée.

C Découpage

Le découpage des lettres dans l'image s'effectue grâce à la détection de chacune d'entre elle dans DETECTION.C.
Effectivement, après avoir établi une liste de toutes les lettres avec les coordonnées attachées à cette-dernière, nous créons une propre fonction à l'intérieur de ce programme avec comme prototype :

```
void cut(SDL_Surface* image, int** list, int listSize);
```

On utilise dans cette méthode l'image de base après prétraitement, la liste de toutes les lettres avec leurs positions (x, y) mais aussi listSize correspondant au nombre d'éléments dans la liste.

Nous établissons à l'intérieur du programme la création d'une dossier permettant de stocker toutes les images qu'on nommera "letters_extracted", développé de cette manière :

```
system("mkdir -p letters_extracted")
```

Suite à cela, nous récupérons toutes les données que nous souhaitons, en l'occurence, toutes les positions de chaque lettre présente.

Grâce à une boucle, les coordonnées sont utilisées de tel manière à créer un rectangle, représentant la cellule, que nous allons récupérer et associée à une surface :

```
SDL_Rect cell_rect = { x_pos, y_pos, wwidth, wheight };
SDL_Surface *cell_surface =
SDL_CreateRGBSurface(0, wwidth, wheight,image->format->BitsPerPixel,
    image->format->Rmask, image->format->Gmask,
    image->format->Bmask, image->format->Amask);
```

Pour conclure cette partie, nous utilisons SDLBlitSurface, nous permettant d'obtenir l'image du découpage de la lettre, résultat obtenu grâce au transfert du rectangle vers la surface de la cellule à travers l'image :

```
SDL_BlitSurface(image, &cell_rect, cell_surface, NULL);
```

Ainsi, on enregistre toutes les images dans le dossier "letters_extracted" avec un format unique (letter_x_y.png) et on libère la mémoire de la surface de la cellule pour terminer :

```
    char filename[listSize];

    snprintf(filename, sizeof(filename),
    "letters_extracted/letter%d%d.png", x_pos, y_pos);

    IMG_SavePNG(cell_surface, filename);
    SDL_FreeSurface(cell_surface);
```

D Solver

Le projet étant de créer un logiciel de résolution de mots caché, il est bien entendu essentiel d'avoir un algorithme de résolution de mots cachés fonctionnel, qui est situé dans le fichier SOLVER.C.

Après compilation, l'exécutable solver peut être appelé directement depuis un terminal via la commande :

```
./solver <nom de fichier> <mot a chercher>
```

Comme on peut le constater, ce programme prend deux arguments : le chemin vers un fichier contenant la grille de mots cachés sous forme de texte, ainsi que le mot à chercher dans cette grille.

Le programme SOLVER est capable de chercher un mot horizontalement, verticalement et en diagonale, le tout de gauche à droite et de droite à gauche. Si le mot à été trouvé, le programme va renvoyer dans la sortie standard un message de type :

```
(x1,y1)(x2,y2)
```

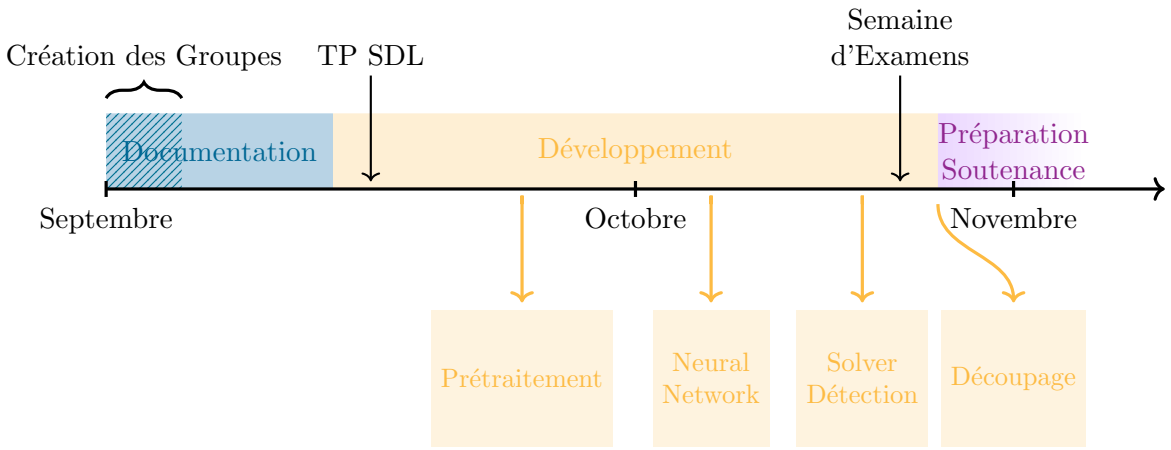
Où (X1,Y1) et (X2,Y2) correspondent respectivement aux coordonnées de début et de fin du mot cherché, (0,0) correspondant au coin supérieur gauche de la grille. En revanche, si le mot n'a pas été trouvé, le programme retourne :

```
Not Found
```

Le programme effectue également une vérification des entrées, en vérifiant que le fichier contenant la grille existe, qu'il contient bien une grille (sans trous) et que les grille possède au moins 5x5 cases. Dans le cas contraire, le programme renvoie une erreur correspondante.

5 État d'avancement du projet

A Feuille de route



B Avancement général des tâches

Nous avons pour cette première soutenance bien progressé, avancé, sur de nombreux points ;

+ Concernant le réseau de neurones, il est fonctionnel et clair. Le code implémenté pour la porte NXOR sera très utile, la plupart des fonctions pourront être réemployées pour la soutenance suivante et pour notre objectif final du projet, en effet les calculs ne changeront points, mais le nombre d'entrées et de sorties pourra varier.
  Il est toujours possible d'optimiser le nombre d'itérations nécessaires à l'apprentissage avec notre taux de précision, en modifiant certains coefficients, valeurs fixes, nombres de neurones cachées.
  Dans l'ensemble, le réseau de neurones est bien avancé.

+ Une partie du traitement des images avant de les segmenter est fonctionnelle. On peut citer la possibilité de conversion d'une image en noir et blanc ou en niveau de gris.

C Rotation non-fonctionnelle mais incomplète

Durant cette première partie du projet, nous avons accusé un certain retard concernant la mécanique de rotation. En effet ce retard a été causé par différents facteurs :

+ On peut citer par exemple, le manque de connaissance de certains membres sur SDL mais aussi plus largement en C provoqué en partie par l'apprentissage de langage en même temps que la programmation du projet.

+ On peut aussi citer la volonté de diversifier les moyens pour le traitement d'une image. Grace à ce temps consacrer au développement de nouveaux moyens pour interagir avec une image, on se retrouve avec une meilleur compréhension des notions. En contre partie, la mécanique de rotation n'a pas pu être poussé plus loin pour le moment.

Il nous reste donc pour la prochaine fois, à améliorer ce système en le rendant plus permissif et robuste.

Pour permettre de rattraper ce retard, nous devrons nous plonger plus profondément dans les différents systèmes de rotation et de sauvegarde que propose le langage C.

D Les bonnes avancées sur le "Solver"

Le programme Solver est complètement terminé et fonctionnel, et aucun obstacle majeur n'a été rencontré durant son développement. Comme il s'agit d'une partie essentielle du projet, nous avons fait de notre possible pour s'assurer qu'il soit fonctionnel dès la première soutenance.

De plus le programme Solver est un algorithme de résolution de mots cachés qui est implémenté en C.
Avec une bonne représentation des grilles et une bonne mentalisation des possibles problèmes, nous avons passé un minimum de temps de développement dessus. Ce gain de temps nous a permis d'améliorer d'autres programmes comme la détection ou le découpage des lettres.

E Problème sur la détection

Dans le projet, la détection est sûrement l'un des sujets les plus complexes mais aussi un des plus importants.

Nous avons ainsi développé plusieurs versions pour la détection, dans le but d'optimiser au maximum les performances et d'atteindre les résultats initialement attendus.

Une première version non concluante, nous permettait de découper de façon net et clair que quelques images du niveau 1. À ce moment, toutes les fonctions présentent dans le Prétraitement n'étaient pas implémentées.
Pour pallier à ce retard, nous avions décidé de "Hard-coder" la détection, mais nos premiers résultats n'étaient pas à la hauteur du défi.

Nous nous sommes tourné vers d'autres méthodes comme les histogrammes d'images pour la segmentation. Cependant, nous avons vite été freiné par certains soucis technique par rapport à l'utilisation de certaines bibliothèques.

Aujourd'hui, le projet possède un nouveau processus de détection. On utilise un système de "Clustering" et de "Bounding-Box".

6 Conclusion

Pour conclure sur ce rapport de projet, nous sommes une équipe de quatre membres motivés et intéressés par ce projet. Nous avons avancé sur de nombreux sujets comme le réseau de Neurones ou la résolution des grilles de mots.
Cependant il reste de nombreuses mécaniques à ajouter. Il est certains que nous allons combler le retard sur certains sujets comme la détection des grilles et mots.
Le projet est sur la bonne voie et une bonne lancée grâce à une bonne communication et à un bon planning sur le travail futur à effectuer.