OCR Word Search Solver: Reconnaissance et résolution automatique de grilles de mots cachés

Théophile Serrand, Grégoire Hadji-artinian, Shabithas Harithas, Éric REN

Rapport de soutenance – Semestre 3
EPITA – Année universitaire 2025 / 2026

Encadrant du projet : David Boucher

Table des matières

1 Introduction

Dans le cadre du module de projet S3 à l'EPITA (année universitaire 2024–2025), notre équipe a été chargée de concevoir un programme complet en langage C capable de reconnaître, interpréter et résoudre automatiquement des grilles de mots cachés à partir d'images. Ce projet s'inscrit dans la continuité des enseignements de programmation avancée, d'algorithmique et d'intelligence artificielle, et vise à appliquer de manière concrète les notions de structures de données, de traitement d'image et de réseau de neurones. L'objectif principal du projet est de créer un OCR (Optical Character Recognition) spécialisé dans l'analyse de grilles de lettres. Le logiciel doit être capable de prendre en entrée une image contenant une grille de mots mélangés, de prétraiter cette image pour améliorer sa lisibilité (grayscale, binarisation, redressement, réduction de bruit), puis de segmenter la grille afin d'extraire chaque lettre individuellement. Ces caractères sont ensuite reconnus par un réseau de neurones entraîné, et les mots recherchés sont identifiés grâce à un solver capable de parcourir la grille horizontalement, verticalement et diagonalement. Enfin, une interface graphique permet de visualiser le traitement et le résultat final de la résolution. Ce rapport présente en détail les différentes étapes du développement de notre projet OCR Word Search Solver. Il débute par une présentation des objectifs initiaux, du planning et de l'avancement, puis décrit les choix techniques et algorithmiques mis en œuvre dans chaque module (prétraitement, segmentation, OCR, solver et interface). Nous exposerons ensuite les difficultés rencontrées, les solutions adoptées ainsi que les perspectives d'amélioration envisagées pour la suite du projet.

1.1 Objectif général du projet

L'objectif principal est l'élaboration d'un **OCR (Optical Character Recognition)** spécialisé dans l'analyse de grilles de lettres.

1.1.1 Fonctionnalités du logiciel

Le logiciel doit opérer en suivant les étapes clés du pipeline de traitement d'images :

- **Prétraitement** : Amélioration de la lisibilité de l'image (niveaux de gris, binarisation, redressement, réduction de bruit).
- **Segmentation** : Découpage de la grille pour isoler chaque caractère.
- **Reconnaissance des caractères** : Identification des lettres par un réseau de neurones préalablement entraîné.
- **Résolution (Solver)** : Recherche des mots demandés dans la grille, par exploration exhaustive (horizontalement, verticalement et en diagonale).
- **Visualisation** : Affichage du résultat final via une interface graphique (SDL2).

Ce rapport présente en détail les phases de développement, les choix techniques et les défis surmontés pour chaque module.

2 Rôles et contributions de l'équipe

2.1 Shabithas Harithas – Responsable du prétraitement d'image

La mission principale liée à cette partie du projet OCR Word Search Solver a consisté à développer le module de prétraitement des images avant la reconnaissance de caractères. Cette étape vise à améliorer la qualité visuelle des grilles afin de faciliter le découpage et la lecture par le réseau de neurones. Plusieurs fonctionnalités clés ont été implémentées, telles que la conversion des images en niveaux de gris puis en noir et blanc, la suppression du bruit (à l'aide de filtres et d'un renforcement du contraste), ainsi que le redressement automatique de l'image. Ce travail a nécessité la manipulation de bibliothèques de traitement d'image en C et a permis d'approfondir la compréhension de l'importance des étapes intermédiaires dans une chaîne OCR complète.

- **Implémentation** : Conversion en niveaux de gris et binarisation (méthode Otsu), application de filtres de débruitage, et correction automatique de l'inclinaison.
- **Technique** : Maîtrise de la manipulation des images bitmap en C.

2.2 Grégoire Hadji-artinian - Responsable du découpage de l'image

Le rôle attribué dans le cadre de ce projet a porté sur le développement de la partie détection et découpage d'image (fichiers situés dans le dossier detection). Cette composante constitue le maillon central de la chaîne de traitement OCR, assurant la liaison entre le prétraitement (Preprocessing) et la reconnaissance de caractères (OCR). L'objectif de ce travail était de prendre en entrée une image binarisée (matrice de 0 et 1, fournie sous la forme d'un tableau unidimensionnel unsigned char *) et de la segmenter afin de produire trois livrables distincts : une nouvelle image contenant uniquement la grille,

une nouvelle image contenant uniquement la liste de mots,

et deux structures Letters regroupant les images standardisées (50x50 pixels) de chaque caractère, prêtes à être transmises au réseau neuronal du module de reconnaissance — une structure pour les lettres de la grille et une autre pour celles de la liste de mots.

Ce module occupe une place cruciale dans la pipeline, garantissant la cohérence et la qualité des données envoyées à la phase de reconnaissance.

- **Objectif** : Segmenter une image binarisée pour extraire et isoler la grille de lettres et la liste des mots à chercher.
- **Livrables** : Deux images distinctes (grille et liste de mots) et deux structures de données contenant les images de chaque caractère, stan-

dardisées à **50x50 pixels**.

2.3 Théophile Serrand – Responsable du réseau de neurones

La mission associée à cette partie du projet a consisté à concevoir et implémenter le réseau de neurones, composant essentiel du pipeline OCR chargé d'assurer la reconnaissance des caractères extraits après le prétraitement et le découpage. Pour la première soutenance, une preuve de concept a été développée afin de vérifier le bon fonctionnement de l'architecture neuronale avant son application à la reconnaissance de lettres. Cette expérimentation s'est appuyée sur l'apprentissage de la fonction logique XNOR, un cas classique démontrant la capacité d'un réseau à modéliser une relation non linéaire nécessitant une couche cachée. Le réseau développé se compose d'une couche d'entrée, d'une couche cachée et d'une couche de sortie. L'apprentissage repose sur la descente de gradient et la rétropropagation de l'erreur, permettant d'ajuster progressivement les poids synaptiques pour minimiser l'erreur moyenne entre la sortie prédite et la sortie attendue. Les résultats obtenus confirment la convergence du modèle, l'erreur moyenne diminuant régulièrement de 0.145455 à 0.000172 après 9000 époques. Cette preuve de concept valide le bon fonctionnement de l'algorithme et servira de base à l'apprentissage des caractères pour la suite du projet.

2.3.1 Preuve de concept XNOR

Une preuve de concept basée sur la **fonction logique XNOR** a été réalisée pour valider le modèle d'apprentissage.

- **Architecture** : Un réseau Perceptron Multi-Couche minimaliste (couche d'entrée, cachée, sortie).
- **Mécanisme** : Utilisation de la **descente de gradient** et de la **rétropropagation** de l'erreur.
- **Résultat** : Convergence démontrée avec une chute de l'erreur moyenne de 0.145455 à 0.000172 après 10000 époques.

2.4 Éric Ren - Responsable du solver

La mission principale associée à cette composante a porté sur la conception et l'implémentation du solver, c'est-à-dire le module chargé d'identifier les mots au sein d'une grille de lettres issue du processus de reconnaissance. Ce module constitue la dernière étape logique du pipeline avant l'affichage des résultats. Le solver reçoit en entrée une matrice de lettres ainsi qu'un mot à rechercher. Son rôle est d'explorer la grille dans l'ensemble des directions possibles, horizontales, verticales et diagonales, afin de déterminer si le mot est présent. En cas de succès, il retourne les coordonnées précises de la première et de la dernière lettre du mot détecté.Ce travail a permis d'assurer la fiabilité et la rapidité de la phase de résolution, garantissant une exécution efficace du programme dans sa version actuelle.

2.4.1 Rôle et Fonctionnalités

— **Input** : Une matrice de lettres (obtenue par l'OCR) et le mot à localiser.
— **Algorithme** : Exploration exhaustive de la grille dans les **huit directions** (horizontale, verticale et diagonale).
— **Output** : Retour des coordonnées précises de début et de fin du mot détecté.

3 Objectifs, planification et bilan d'avancement

3.1 Objectifs généraux

L'objectif est d'aboutir à un programme intégré combinant traitement d'image, segmentation, OCR et résolution algorithmique en **langage C**.

3.2 Planification prévisionnelle

Phase	**Objectifs principaux**	**Période prévue**	**État d'avancement**
Phase 1	Prétraitement : Binarisation, rotation, débruitage	16 septembre → 31 octobre	Terminée
Phase 2	Découpage : Détection de grille et des lettres	16 septembre → 31 octobre	Terminée
Phase 3	OCR : Mise en place du réseau (preuve XNOR)	16 septembre → 31 octobre	Terminée
Phase 4	Solver : Algorithme de recherche de mots	16 septembre → 31 octobre	Terminée
Phase 5	Interface graphique (SDL) : Affichage et interaction	31 octobre → 12 décembre	10%
Phase 6	Site web du projet : Documentation et présentation	20 octobre → 15 novembre	50%

3.3 Bilan d'avancement

Le projet est **solide et conforme à la planification**. Les fondations techniques sont posées : le prétraitement, la segmentation, le solver et la preuve de concept du réseau de neurones sont fonctionnels et intégrés.

3.3.1 Prochaines étapes

Les efforts se concentreront sur la finalisation du **réseau OCR** (apprentissage sur des caractères réels) et le développement complet de l'**interface graphique SDL2**.

4 Architecture technique et choix algorithmiques

4.1 Prétraitement de l'image

Le module de prétraitement constitue la première étape essentielle du projet OCR Word Search Solver. Son objectif principal est de transformer une image brute, souvent issue d'une photographie, en une image propre, contrastée, redressée et exploitable par la suite du programme. L'ensemble des opérations menées vise à réduire le bruit, homogénéiser les couleurs et corriger les imperfections liées à la prise de vue, de manière à fournir une base stable pour la détection des lettres et la reconnaissance optique de caractères (OCR). Ce module a été développé intégralement en langage C, en s'appuyant principalement sur la bibliothèque SDL2 (Simple DirectMedia Layer) et son extension SDL2-image, utilisées pour le chargement, la manipulation et la sauvegarde des images bitmap. L'architecture du prétraitement repose sur plusieurs fichiers distincts : color-modif.c pour la conversion en niveaux de gris, binarisation.c pour le seuillage adaptatif, rotation.c pour la correction d'inclinaison (manuelle et automatique), cleaner.c pour la réduction du bruit, et preprocessing.c pour l'orchestration générale des étapes et la sauvegarde des résultats intermédiaires. Ce découpage modulaire facilite les tests unitaires et favorise la réutilisation des fonctions dans les modules suivants (segmentation, OCR, solver). Le processus débute par la suppression des couleurs, une étape indispensable convertissant l'image en niveaux de gris. Cette pondération, qui reflète la sensibilité de l'œil humain à la lumière, élimine toute information chromatique inutile. La binarisation est ensuite appliquée selon la méthode d'Otsu, qui détermine automatiquement le seuil optimal séparant le fond (blanc) du texte (noir) en maximisant la variance interclasse. Ce procédé présente l'avantage d'être entièrement automatique et robuste face aux variations de luminosité, produisant une image strictement binaire dans laquelle les lettres ressortent nettement. La correction d'inclinaison constitue l'étape suivante. Deux approches ont été mises en œuvre : une rotation manuelle et une rotation automatique. La première, exigée pour la soutenance intermédiaire, permet à l'utilisateur d'entrer un angle de redressement directement depuis le terminal, assurant un contrôle précis et visuel du résultat. La seconde, déjà implémentée en vue de la soutenance finale, repose sur une analyse de variance par balayage angulaire (« ray casting ») : pour différents angles, le programme calcule la variance des densités de pixels noirs le long des lignes horizontales, l'angle présentant la variance maximale étant considéré comme celui où le texte est le plus droit. L'image est ensuite redressée pixel par pixel à l'aide des fonctions trigonométriques cos() et sin(). Enfin, une réduction du bruit est appliquée afin d'éliminer les imperfections restantes. L'algorithme repose sur une analyse locale du voisinage de chaque pixel : si la majorité des pixels environnants sont blancs, celui-ci est également converti en

blanc. Ce filtrage binaire supprime efficacement les artefacts de rotation et les points isolés, tout en préservant la structure principale des caractères. Les différentes étapes génèrent plusieurs images intermédiaires enregistrées dans le dossier output/ : image-grayscale.bmp (conversion en niveaux de gris), image-binarize.bmp (binarisation), image-auto-rotation.bmp (rotation automatique), image-manual-rotation.bmp (rotation manuelle), image-noise-reduc-auto.bmp et image-noise-reduc-manual.bmp (image finale). Pour la première soutenance, la rotation manuelle constitue la version officiellement évaluée, tandis que la rotation automatique est conservée pour la phase finale du projet. Le choix du langage C s'impose pour son contrôle précis de la mémoire et ses performances dans le traitement pixel par pixel. La bibliothèque SDL2 offre une interface simple et portable pour la manipulation directe des surfaces. L'organisation modulaire du code favorise la clarté, la maintenabilité et la testabilité. L'ensemble du module est compilé avec les options -Wall et -Wextra, garantissant la rigueur du code et l'absence de comportements indéfinis. Ce module constitue ainsi une base stable et efficace pour les étapes suivantes de détection et de reconnaissance des caractères.

4.2 Segmentation de l'image

Le module de découpage a été développé intégralement en langage C, sans recours à des bibliothèques externes de traitement d'image telles qu'OpenCV ou SDL. L'ensemble du travail repose exclusivement sur les bibliothèques standards du C : stdio.h, pour les fonctions de débogage et d'affichage de statut (printf, fprintf) ;

stdlib.h, essentielle pour la gestion dynamique de la mémoire (malloc, calloc, free) nécessaire à la création des nouvelles images de lettres.

Tous les algorithmes de manipulation d'image (projections, copie de pixels, centrage) ont été implémentés manuellement à l'aide de ces outils. La structure Image, stockée dans un tableau unidimensionnel (unsigned char *data), a représenté un défi particulier, l'analyse devant être effectuée en deux dimensions. La conversion d'indices a donc été gérée explicitement selon la formule data[y * width + x], garantissant la cohérence du parcours des pixels. Plusieurs approches ont été explorées avant d'aboutir à la solution finale. Une méthode initiale supposant une disposition fixe (grille en haut, mots en bas) s'est révélée trop fragile face à la variabilité des images. La solution retenue repose sur une analyse par projection et détection de blobs. Cette technique consiste à calculer les sommes de pixels par ligne et par colonne afin d'identifier les zones denses (grille, liste de mots). La fonction find-first-blob() détecte la boîte englobante du premier bloc, le copie dans une nouvelle image, puis l'efface pour détecter le bloc suivant. Cette approche dite "destructive" assure une robustesse face au bruit et à la disposition variable des éléments. Enfin, les lettres extraites sont normalisées dans des images de taille fixe (50×50 pixels) sans déformation, afin de préserver leurs proportions pour le réseau de neurones. Ce module produit donc plusieurs livrables : une image

contenant uniquement la grille, une autre contenant la liste de mots, ainsi que deux structures Letters regroupant les caractères extraits.

4.3 Réseau de neurones

Le module de réseau de neurones implémente un Perceptron Multi-Couche (MLP) minimaliste, développé intégralement en langage C. Ce choix d'implémentation bas niveau permet une maîtrise complète du calcul matriciel et du processus d'optimisation, assurant une compréhension approfondie des mécanismes d'apprentissage. L'architecture retenue est un réseau à trois couches : une couche d'entrée (2 neurones), une couche cachée (taille ajustée pour gérer la non-linéarité du problème) et une couche de sortie (1 neurone). Cette configuration est spécifiquement conçue pour résoudre le problème non linéaire du XNOR, fonction logique nécessitant impérativement une couche cachée pour être correctement modélisée. Les paramètres principaux (poids, biais et activations) sont stockés dans des tableaux statiques afin d'optimiser les accès mémoire. L'initialisation est effectuée aléatoirement dans l'intervalle [-1, 1], à l'aide de la fonction rand() et d'une graine basée sur l'heure système (srand(time(NULL))), garantissant la rupture de symétrie nécessaire à l'apprentissage. La fonction d'activation sigmoïde o(x) = 1 / (1 + exp(-x)) introduit la non-linéarité indispensable. Sa dérivée, o'(x) = x * (1 - x), est utilisée lors de la rétropropagation du gradient, implémentée selon la règle de descente de gradient pour minimiser l'erreur quadratique moyenne (MSE). L'entraînement, réalisé sur 10 000 époques avec un taux d'apprentissage de 0.5, démontre la convergence du modèle, l'erreur moyenne passant de 0.145455 à 0.000172. Ces résultats confirment la bonne implémentation des mécanismes de propagation avant (forward propagation) et de rétropropagation (backpropagation). Ce module constitue une base solide pour l'étape suivante de reconnaissance de caractères, où le réseau sera entraîné sur un jeu de données d'images de lettres segmentées.

- **Initialisation** : Poids aléatoires dans l'intervalle $[-1, 1]$.
- **Fonction d'activation** : Utilisation de la **sigmoïde** $\sigma(x) = \frac{1}{1+e^{-x}}$ pour l'introduction de la non-linéarité.
- **Optimisation** : Rétropropagation de l'erreur par **descente de gradient** en utilisant la dérivée $\sigma'(x) = x \cdot (1 - x)$.

4.4 Solver

Le module du solver valide la cohérence du pipeline complet : un fonctionnement correct de ce composant confirme la fiabilité des étapes de segmentation et de reconnaissance en amont. Développé intégralement en langage C, le solver s'appuie exclusivement sur les bibliothèques standards : stdio.h pour la lecture et l'affichage des grilles et résultats,

stdlib.h pour la gestion dynamique de la mémoire,

string.h pour la manipulation de chaînes de caractères,

ctype.h pour la normalisation et la conversion des lettres.

L'algorithme explore la grille dans les huit directions possibles (horizontales, verticales et diagonales) afin d'y rechercher un mot donné. Pour chaque case, si la lettre correspond à la première du mot, une vérification est effectuée dans toutes les orientations. Le parcours s'arrête dès qu'une correspondance échoue, optimisant ainsi les performances. Lorsqu'un mot est trouvé, le solver enregistre les coordonnées précises de la première et de la dernière lettre. Le module gère également les mots inversés (de droite à gauche ou du bas vers le haut) et prend en compte les cas particuliers (mots d'une seule lettre, grilles irrégulières, etc.). Les tests réalisés confirment la robustesse de l'algorithme, notamment pour les mots situés en diagonale ou en bordure de grille. La structure du code repose sur des tableaux 2D et des structures de données simples pour stocker les coordonnées et les résultats, garantissant lisibilité et rapidité. La compilation est effectuée avec les options -Wall et -Wextra, assurant la détection des avertissements et la stabilité de l'exécutable. Ce module, entièrement fonctionnel, constitue la dernière étape logique du projet et permettra, à terme, de visualiser les résultats directement sur la grille via le module d'affichage actuellement en développement.

5 Analyse des difficultés et solutions apportées

5.1 Prétraitement de l'image

Le développement du module de prétraitement a représenté une phase particulièrement délicate du projet, en raison de la combinaison de problématiques techniques liées au traitement d'image bas niveau et des contraintes strictes imposées par le langage C. La première difficulté majeure a concerné la manipulation directe des pixels. La bibliothèque SDL2 ne fournissant pas de fonctions de haut niveau pour ce type d'opérations, il a été nécessaire de gérer manuellement les pointeurs, les formats de couleur et les conversions entre canaux RGB. Plusieurs erreurs d'accès mémoire (segmentation faults) sont survenues au début du développement, notamment lors des tests de rotation, en raison d'indices dépassant les dimensions de l'image. Ce problème a été résolu par la mise en place de fonctions utilitaires robustes (image-get-pixel et image-set-pixel) centralisant les vérifications de bornes et la gestion des octets selon le format interne des surfaces SDL. Une autre difficulté importante a concerné la correction d'inclinaison (rotation). L'implémentation d'une rotation géométrique précise nécessite d'effectuer des calculs trigonométriques inverses afin de repositionner correctement chaque pixel. Les premières versions de l'algorithme produisaient des déformations et des bandes noires, dues à une mauvaise gestion des coordonnées et à une inversion entre les axes X et Y. Ces erreurs ont été corrigées après plusieurs visualisations intermédiaires et par l'ajout d'un fond blanc par défaut pour remplir les zones vides. La rotation automatique (deskew) a également posé des difficultés. Certaines grilles peu contrastées induisaient des estimations d'angle incorrectes. Pour stabiliser les résultats, la recherche d'angle a été limitée à

un intervalle restreint [-15 deg, +15 deg], et la binarisation préalable a été améliorée. Ces ajustements ont permis d'obtenir des rotations plus fiables et visuellement cohérentes. Dans son état actuel, le module de prétraitement est pleinement fonctionnel et fournit des images redressées, contrastées et prêtes à être segmentées pour la suite du pipeline OCR.

5.2 Decoupage de l'image

Le développement du module de découpage a nécessité la mise en œuvre de techniques spécifiques de projection et de centrage pour garantir une extraction correcte des lettres sans déformation. Pour préserver le ratio des caractères, plusieurs étapes successives ont été mises en place. Extraction : La lettre est isolée à l'aide de projections successives permettant de détecter les zones contenant du contenu utile.

Standardisation : La fonction standardiser-et-creer-image crée une nouvelle image vide de 50×50 pixels, initialisée à zéro grâce à calloc.

Centrage : Les marges horizontales et verticales nécessaires au centrage de la lettre sont calculées, puis les pixels de la lettre extraite sont copiés dans le nouveau cadre selon le décalage déterminé.

La partie détection est désormais fonctionnelle et remplit ses trois objectifs principaux : produire une image contenant uniquement la grille (img-grille),

une image contenant uniquement la liste de mots (img-mots),

et deux structures de type Letters contenant les lettres extraites, centrées et standardisées au format 50×50 pixels, prêtes à être transmises au réseau neuronal pour la phase de reconnaissance.

Ce module garantit la qualité et la cohérence des images d'entrée du réseau de neurones, assurant une transition fluide entre la segmentation et l'apprentissage.

5.3 Réseau de neurones

Le développement du module de réseau de neurones a présenté plusieurs défis techniques liés à la précision numérique et à la stabilité de l'apprentissage. Toute erreur dans l'alignement des indices lors de la rétropropagation conduit à une corruption de la formule du gradient, provoquant soit une divergence, soit une incapacité à minimiser l'erreur. Le réglage des hyperparamètres s'est également révélé crucial : un taux d'apprentissage trop élevé entraînait des oscillations autour du minimum de la fonction de perte, tandis qu'un taux trop faible ralentissait considérablement la convergence. L'initialisation des poids a constitué un autre point sensible. Des valeurs initiales inadéquates pouvaient saturer la fonction sigmoïde (problème de vanishing gradient), annulant les gradients et empêchant l'apprentissage de certains neurones. L'utilisation d'une initialisation centrée sur zéro dans l'intervalle [-1, 1] a permis de limiter ce risque. La vérification de la convergence a été effectuée par l'affichage régulier de l'erreur moyenne toutes les 1000 époques,

confirmant la justesse de l'implémentation. La fonction tester() valide la capacité du modèle à séparer correctement les classes non linéaires du problème XNOR, avec des sorties binaires prédictes correctement après seuillage à 0.5. Le code C obtenu constitue un livrable fonctionnel démontrant la maîtrise des fondements algorithmiques de l'apprentissage profond. Bien qu'il s'agisse pour l'instant d'une version de test, l'architecture est complète et pourra être reliée au reste du pipeline afin d'identifier des lettres issues des images traitées. Les images des lettres extraites pourront alors être transformées en caractères, permettant la construction de deux matrices : l'une pour la grille et l'autre pour la liste des mots à rechercher. Ce module assure ainsi la transition entre le traitement d'image et la reconnaissance symbolique, étape clé du projet OCR.

5.4 Solver

Le développement du module solver a nécessité une attention particulière à la rigueur algorithmique, notamment dans la gestion des indices et de la mémoire. Une erreur de calcul de coordonnées ou un dépassement de tableau pouvait compromettre la stabilité du programme. Des vérifications systématiques ont donc été intégrées avant chaque accès mémoire, et le code a été testé sur des grilles de différentes tailles afin de garantir sa robustesse. La validation des résultats a également posé des défis. Lors des premiers tests, certains mots n'étaient pas détectés en raison de différences de casse ou de la présence d'espaces parasites. L'ajout d'une phase de normalisation (conversion en majuscules et suppression des caractères non pertinents) a permis d'éliminer ces erreurs. Des problèmes de performance ont également été observés sur des grilles de grande dimension. Une optimisation a été introduite en interrompant la recherche dès qu'une lettre ne correspondait plus au mot cible, réduisant ainsi le nombre de comparaisons inutiles et améliorant significativement le temps d'exécution. Le solver gère désormais la recherche dans les huit directions possibles (horizontales, verticales et diagonales), y compris les mots inversés. L'algorithme repose sur des tableaux 2D pour représenter la grille et sur des structures simples pour stocker les coordonnées des mots trouvés. L'ensemble du module a été développé en langage C, à l'aide des bibliothèques standards (stdio.h, stdlib.h, string.h, ctype.h). Ces choix garantissent la portabilité, la légèreté et la compatibilité complète avec les autres modules du projet. Ce composant, stable et testé, valide la cohérence globale du pipeline. Il constitue la dernière étape du traitement, assurant la recherche fiable des mots et confirmant la bonne intégration entre la segmentation, la reconnaissance et la résolution.

6 Conclusion et perspectives

Au terme de cette première phase du projet, nous avons atteint un niveau de développement solide sur les fondations du système OCR appliqué à la résolution de grilles de mots cachés. Le prétraitement des images est désor-

mais entièrement opérationnel : nous avons implémenté un pipeline complet permettant de transformer une image brute en une version nette, binarisée et redressée. Cette étape comprend la conversion en niveaux de gris, la binarisation automatique via la méthode d'Otsu, le redressement manuel et automatique afin de corriger les inclinaisons de la grille, ainsi qu'une réduction de bruit pour éliminer les pixels parasites. Ces traitements garantissent que les images obtenues sont propres et exploitables pour la suite du processus. Nous avons également mis en place le découpage (segmentation) des lettres, permettant d'extraire chaque caractère de la grille. Cette étape constitue un pont essentiel entre le prétraitement et l'OCR, puisqu'elle permet de générer les sous-images qui serviront d'entrées au réseau de neurones. Grâce à ce module, les lettres détectées peuvent être isolées, sauvegardées et ensuite envoyées dans le réseau pour apprentissage ou reconnaissance. Concernant le réseau de neurones, nous avons réalisé une preuve de concept fonctionnelle basée sur la fonction logique XNOR. Cette version simplifiée nous a permis de valider l'implémentation mathématique du réseau en C, notamment la propagation avant, la rétropropagation et la mise à jour des poids. Cette première expérience nous assure que la structure du réseau et les opérations de base sont correctes avant de passer à une version entraînée sur des lettres réelles. Même si certaines parties du projet ne sont pas encore finalisées, comme l'affichage graphique complet ou la reconnaissance OCR finale, les modules essentiels sont déjà en place et communiquent correctement entre eux. Le projet dispose d'une architecture claire et modulaire (UTILS, Preprocessing, Detection, OCR, Solver, etc.) et d'un Makefile fonctionnel, facilitant la compilation et l'intégration continue. Les images intermédiaires générées (grayscale, binarisation, deskew, noise reduction, segmentation) permettent de visualiser les progrès du pipeline, même en l'absence d'une interface graphique. En conclusion, cette première soutenance marque une avancée significative : le cœur du traitement d'image et du réseau est en place, la segmentation fonctionne, et les bases de l'OCR sont posées. Pour la seconde soutenance, nos objectifs sont clairs, finaliser le réseau de neurones complet pour la reconnaissance de lettres,intégrer un affichage graphique interactif et améliorer la précision et la rapidité du pipeline.

7 Références bibliographiques

Références

— Wikipedia contributors. (2025). **Image Noise Reduction**. In *Wikipedia, The Free Encyclopedia*. Disponible sur : `https://en.wikipedia.org/wiki/Image_noise_reduction`
↪ Référence pour les techniques de réduction de bruit basées sur l'analyse locale (voisinage de pixels).

— OpenCV Documentation. (2024). **Image Thresholding and Rota-**

tion. *OpenCV Official Docs*. Disponible sur : `https://docs.opencv.org/`
↪ Consulté pour la compréhension des approches modernes de rotation d'image et de deskew automatique via des transformations géométriques.

— OpenCV Documentation. (2024). **Image Thresholding, Filtering, and Geometry Transformations**. Disponible sur : `https://docs.opencv.org/`
↪ Source de documentation pratique pour la mise en œuvre de la binarisation Otsu et du redressement automatique via `cv::threshold` et `cv::getRotationMatrix2D()`.

— SDL2 Documentation. (2024). **Simple DirectMedia Layer – Pixel Access and Image Handling**. Disponible sur : `https://wiki.libsdl.org/`
↪ Référence technique pour la manipulation bas-niveau des pixels en C, utilisée pour coder les fonctions `image_get_pixel()` et `image_set_pixel()`.

— SDL2 Library Documentation. (2024). **Simple DirectMedia Layer – Image Processing**. Disponible sur : `https://wiki.libsdl.org/`
↪ Documentation utilisée pour l'implémentation des fonctions de manipulation d'images (`SDL_GetRGB`, `SDL_MapRGB`, `SDL_SaveBMP`), essentielles au traitement et à la sauvegarde des images.

— C Standard Library. (N/A). **math.h – Fonctions mathématiques**.
↪ Librairie standard du C, cruciale pour les calculs fondamentaux du réseau de neurones, notamment `exp()` (pour l'activation Sigmoïde) et `pow()` (pour la fonction de coût de l'erreur quadratique).

— C Standard Library. (N/A). **time.h – Utilitaires de date et heure**.
↪ Librairie standard du C, utilisée pour l'initialisation non déterministe du générateur de nombres aléatoires (`srand(time(NULL))`), assurant une initialisation variée des poids et biais du réseau de neurones.

— cppreference.com. (2025). **C Standard Library Reference**. Disponible sur : `https://en.cppreference.com/w/c`
↪ Documentation de référence pour les fonctions standards du C, notamment la gestion mémoire, les chaînes de caractères et les entrées/sorties.

— GeeksforGeeks. (2024). **Search a Word in a 2D Grid of Characters**. Disponible sur : `https://www.geeksforgeeks.org/search-a-word-in-a-2d-grid-of-`
↪ Source consultée pour la conception de l'algorithme de recherche de mots dans une grille à huit directions.

— Stack Overflow Contributors. (2025). **Handling Arrays and Pointers in C – Best Practices**. Disponible sur : `https://stackoverflow.com/`
↪ Référence utilisée pour les bonnes pratiques de manipulation des tableaux et des pointeurs en C.