

Rapport de dernière soutenance

Vladimir Meshcheriakov Eva Blum
Yves-Antoine Gangner Vincent Massard

6 Décembre 2021



Les SpeeedyCuber présente
l'OCR

Table des matières

1	Introduction	3
1.1	Qu'est qu'un OCR ?	3
1.2	Les étapes à suivre	4
2	Présentation du groupe	5
2.1	Vladimir Meshcheriakov	5
2.2	Eva Blum	5
2.3	Yves-Antoine Gangner	5
2.4	Vincent Massard	6
2.5	Organisation	6
3	Résolution du Sudoku	7
3.1	Algorithme du solver	7
4	Prétraitement de l'image	9
4.1	Chargement de l'image	9
4.2	Nuance de gris	10
4.3	Binarisation	11
4.4	Élimination du bruit	13
5	Segmentation et rotation	20
5.1	Préparation de la segmentation	20
5.2	Détection de lignes et colonnes	23
5.3	Détection de l'angle	27
5.4	Rotation de l'image	28
5.5	Segmentation	30
6	Réseau de neurones	32
6.1	Introduction	32
6.2	Qu'est ce que c'est ?	32

6.3	Implémentation	33
6.4	Apprentissage	35
6.5	La porte XOR	36
7	Interface graphique	36
7.1	Librairie GTK	36
7.2	Notions de base	37
7.3	Les box	38
7.4	Les widget	38
7.5	Les onglets	39
7.6	Caractérisation de notre interface graphique . . .	39
8	Libération de la mémoire et séparation des fichiers	41
9	Site internet	42
9.1	La réflexion	42
9.2	L'implémentation	43
10	Conclusion	45

1 Introduction

Comme pour toute année à l'EPITA, nous avons à réaliser un projet de groupe. Là où l'année dernière nous avons dû nous frotter à la difficile tâche de réaliser un jeu vidéo dans son intégralité, nous allons devoir faire face à la réalisation d'un logiciel de Reconnaissance Optique de Caractères plus couramment appeler OCR (en anglais : Optical Character Recognition).

Nous pensons que ce projet nous sera très bénéfique car il nous fera utiliser le langage C dans toutes ses fonctionnalités notamment les pointeurs et la gestion de la mémoire. Ceci, nous obligera à une certaine rigueur que nous n'avons pas encore totalement. De plus, ce projet nous permettra également de nous initier au domaine de la reconnaissance d'images ainsi que tous les principes mathématiques qui y sont liés comme la transformée de Hough.

Nous avons choisi de nommer ce projet "Speedy Cuber" car nous avons pour ambition de résoudre le sudoku donné par un utilisateur le plus rapidement possible.

1.1 Qu'est qu'un OCR ?

Un OCR (Optical Character Recognition) est un logiciel qui a pour but de traduire une image scannée en un fichier de texte. Autrement dit, le contenu de l'image doit pouvoir être utilisable, ce qui n'est pas le cas sur un fichier image.

1.2 Les étapes à suivre

Le fonctionnement d'un OCR se déroule en plusieurs étapes distinctes.

Dans un premier temps, il faut charger l'image souhaité.

Dans un second temps, il y a le prétraitement de l'image. Cette étape est essentielle au bon déroulement du reste du programme car le but est d'améliorer éventuellement la qualité de l'image. Ceci peut inclure le redressement d'images inclinées ou déformées, des corrections de contraste, supprimer des couleurs, c'est-à-dire garder uniquement des niveaux de gris puis le passage en mode bicolore (noir et blanc) et enfin la détection de contours.

Dans un troisième temps, il y a l'étape de la détection de la grille ainsi que la détection des cases donc c'est la segmentation en lignes et en caractères (ou analyse de page). Elle a pour but d'isoler dans l'image les lignes de la grille et les caractères à l'intérieur des lignes pour ensuite récupérer ces derniers.

Puis, l'étape de la reconnaissance des caractères qui est, de loin, la plus ardue. Elle nécessite l'utilisation d'un réseau de neurones qui servira d'identificateur de caractères mais aussi apprendre de ses erreurs pour devenir plus performant.

Enfin, dans un dernier temps, nous devrons gérer la reconstruction de la grille, la résolution du sudoku, son affichage et pour finir, sa sauvegarde.

Ici, notre sujet porte sur le résolution d'un sudoku de dimension 9x9.

2 Présentation du groupe

2.1 Vladimir Meshcheriakov

Le C est un langage super important pour n'importe quel informaticien. Je suis donc ravi de faire partie de mon groupe cette année, pour améliorer mes connaissances dans ce domaine. L'OCR est un challenge qui vaut les nuits sans sommeils pour parvenir à un résultat attendu.

2.2 Eva Blum

Dès mon plus jeune âge, j'ai toujours été passionnée par le monde de l'informatique qui est si vaste et sans limites. Après une année passée à EPITA, mener des projets de bout à bout est très satisfaisant et nous permet de grandement progresser. L'OCR est un projet qui m'intéresse beaucoup car il comprends différentes parties comme le réseau de neurones et le traitement de l'image. Ces parties m'intriguaient beaucoup depuis ma rentrée à EPITA donc je suis très heureuse de faire ce projet accompagnée de mes camarades.

2.3 Yves-Antoine Gangner

Je suis passionné d'informatique et adore principalement le python, mais je suis tout autant intéressé par le C et souhaite en apprendre d'avantage sur ce langage comme sur les IA. Le projet est une occasion pour moi de travailler et solidifier des éléments tels que la cohésion d'équipe, l'organisation de groupe ou bien ce que peut impliquer un projet sur une période de temps défini.

2.4 Vincent Massard

Depuis quelques années, le milieu de l'informatique m'attire et les différentes technologies qui évoluent au cours du temps me fascinent au plus haut point. Cet intérêt ne fait que grandir à force d'expérimentation, d'heure de code, de perte de sommeil et bien entendu d'échec cuissant, sans pour autant abandonné, je trouve toujours intéressant et pratique de réellement se documenter pour un projet et réutiliser ces compétences dans d'autres contextes. J'attends entre autre de l'OCR se même engouement.

2.5 Organisation

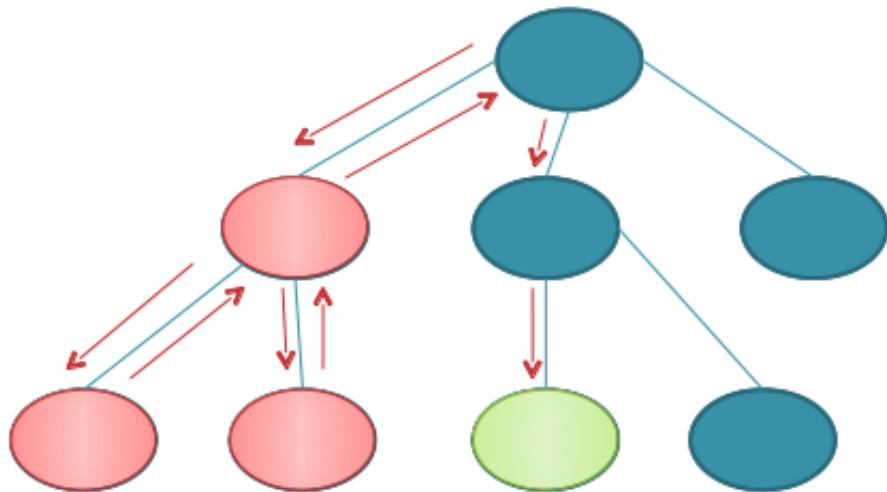
	Eva	Vladimir	Vincent	Yves-Antoine
Pré-traitment	x	x		
Segmentation		x		
Solver du sudoku			x	
Interface graphique			x	
XOR				x
Réseau de neurones				x
Rotation de l'image	x			
Découpage de l'image		x		
Site internet	x			

3 Résolution du Sudoku

3.1 Algorithme du solver

Comme précédemment explicité, notre projet consiste à la lecture d'une image, la détection d'un grille de sudoku, ses différentes cases (vides ou non), et bien évidemment sa résolution. De manière simple, nous avons opté pour une méthode récursive (ou de backtracking) ; Le backtracking est une forme de parcours en profondeur d'un arbre avec des contraintes sur les noeuds.

L'idée est de partir du noeud parent, descendre dans le premier noeud fils satisfaisant la contrainte. Ce noeud fils devient alors un noeud parent et l'on parcourt ensuite ses noeuds fils sous le même principe. Lorsque l'on a parcouru tous les noeuds fils d'un noeud et qu'aucun ne satisfait la contrainte, on remonte alors au noeud parent et on descend dans le noeud fils suivant. Si l'on arrive au dernier fils du premier noeud parent et qu'il ne satisfait pas la contrainte alors il n'existe pas de solution. La solution est identifiée lorsque l'on arrive à un noeud qui satisfait la contrainte et qui n'a pas de noeud fils.



Ainsi on peut se concentrer sur les éléments importants de l'algorithme de résolution. On classe les cases de celles ayant le moins de possibilités à celles en ayant le plus.

On place ce classement dans une liste. On parcours la liste jusqu'à arriver à la dernière cellule de la liste.

Pour chaque cellule de la liste :

- On teste les valeurs de 1 à n^2 :
- si la valeur est possible :
 - on l'inscrit dans la cellule et on passe à la suivante
 - sinon :
 - on remonte à la cellule suivante et on reprend le test des valeurs de 1 à n^2 à partir de la valeur déjà inscrite dans la cellule.
 - dans le cas où aucune valeur correspondante n'est trouvé, on considère le sudoku insoluble.

4 Prétraitement de l'image

4.1 Chargement de l'image

Notre programme se basant sur une ou plusieurs images en entrée. La première chose à faire était donc de réussir à charger une image afin de l'exploiter. Pour cela, nous exploitons la librairie SDL. En utilisant SDL, nous avons accès à chaque pixel d'une image sous la forme d'un tableau à deux dimensions. Par exemple la fonction `SDL_SaveBMP` nous permet de sauvegarder une image sous le format de BMP pour ensuite manipuler nos images. La justification de l'utilisation de ce format, est assez simple : nous avons voulu simplifier la manipulation d'une image et BMP est un parfait format pour débuter.

4.2 Nuance de gris

Avant tout traitement, il faut transformer l'image en noir et blanc ce qui nous permettra de récupérer plus facilement les caractères.

Mais dans un premier temps, il est nécessaire de passer en niveau de gris. Pour cela, nous nous sommes basés sur ce que nous avons vu en TP en début d'année. Pour rappel on calcule la luminosité d'un pixel grâce à la fonction suivante :

$$\text{Luminosité} = 0.3 * r + 0.59 * g + 0.11 * b$$

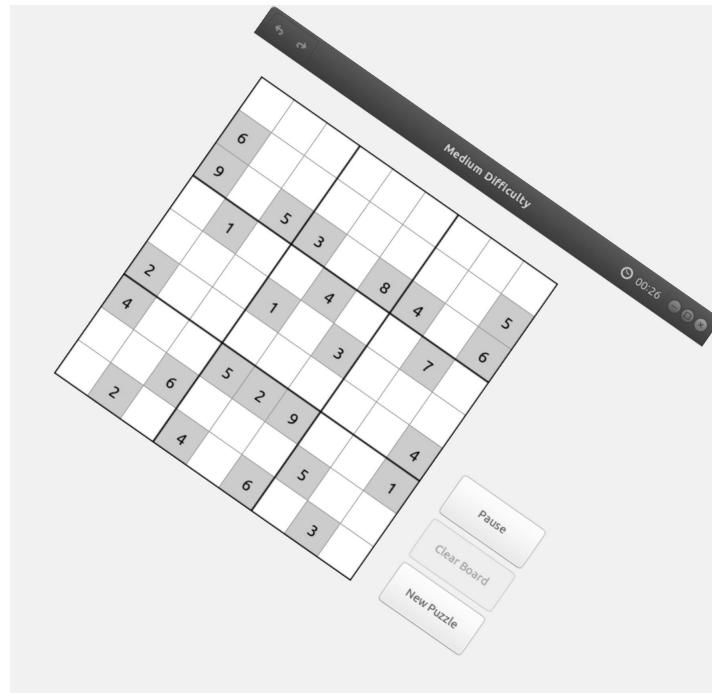


FIGURE 1 – Démonstration du greyscale

Grâce au fait que les pixels de notre image sont ramenés d'un triplé de 8 bits vers une unique valeur de 8 bits, on réduit suffisamment l'information de l'image pour ne pas allourdir les calculs.

4.3 Binarisation

Il existe un nombre conséquent d'algorithmes de binarisation. Mais nous avons essayé d'implémenter ceux qui nous ont semblé les plus efficaces et optimisés. Par exemple, la binarisation de Otsu qui parcourt l'image et établit un histogramme, dans lequel il va chercher un seuil. Celui-ci fera la distinction claire entre l'arrière plan de l'image et son premier plan. Ce dernier a donc pour but de souligner davantage les objets exposés en premier plan.

Néanmoins Otsu ne donne pas toujours de très bons résultats. On fait donc recours à une autre technique qui est un seuil adaptable à chaque image. Il donne donc généralement de meilleurs résultats que Otsu mais est plus coûteux en compilation. L'algorithme parcourt des zones de l'image et adapte un seuil à chaque zone.

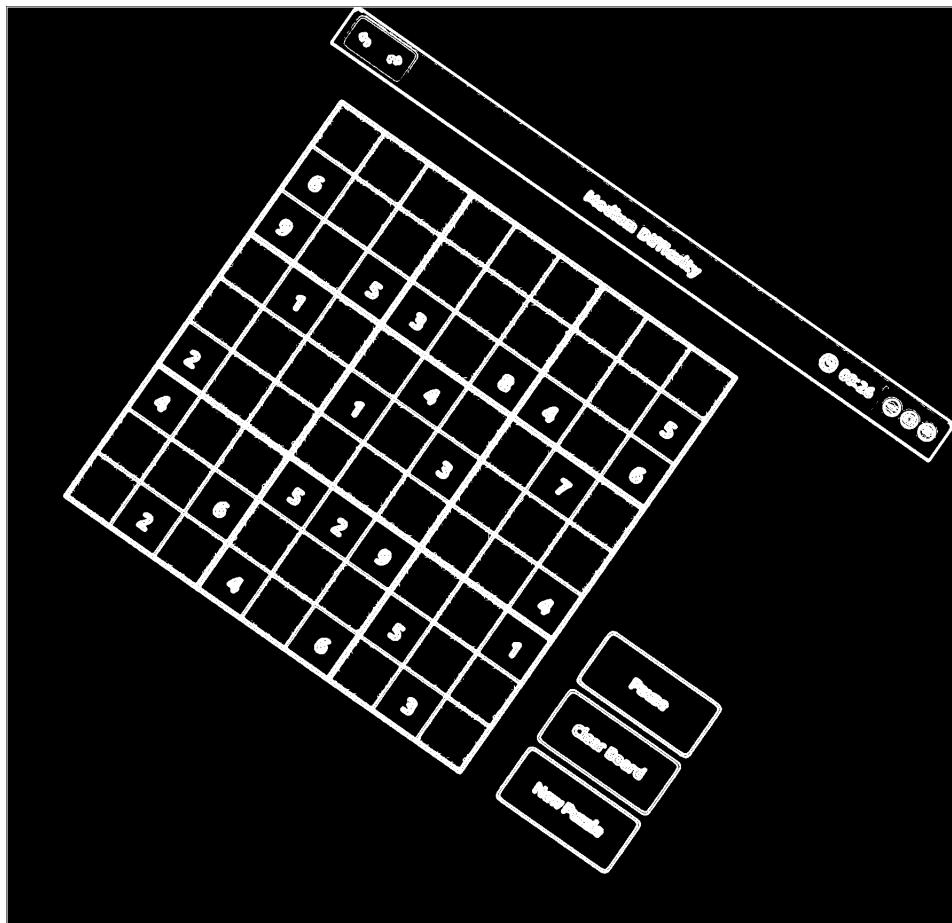


FIGURE 2 – Démonstration de la binarisation

4.4 Élimination du bruit

Le bruit d'une image est le nom donné à la présence d'informations parasites qui viennent s'ajouter de façon aléatoire aux détails d'une image numérique. La principale conséquence de cela est la perte de netteté de l'image dû à l'apparition de pixel parasite nommé artefact.

Comme tout logiciel permettant un traitement d'image, nous nous sommes rendu compte de la nécessité d'utiliser des filtres. Nous avons, dans un premier temps, concentré nos efforts sur un filtre permettant d'éliminer les pixels parasites (créés par la binarisation et la rotation) puis nous avons également créé un filtre de contraste pour contrebalancer le flou créé par le filtre précédent.

Ainsi après avoir effectué des recherches nous avons trouvé des méthodes différentes pour éliminer le bruit : le threshold, la transformation par histogramme, le sobel, l'égalisation d'histogramme, le filtre médian, le gaussien, la méthode d'Otsu. Bien évidemment ces différents filtres ont pour but d'éliminer les artefacts. Nous avons donc décidé d'implémenter ces types de filtres pour pouvoir choisir le plus adapté pour la première soutenance. Pour la seconde soutenance, nous avons donc trouvé le parfait ordre de superposition des filtres pour avoir une qualité d'image suffisante pour l'analyser par la suite. Il s'agit d'appliquer le threshold, la transformation par histogramme, le filtre de Sobel et enfin l'égalisation d'histogramme.

Le threshold :

L'histogramme est parfois très utile pour segmenter l'image en deux classes, c'est-à-dire pour distinguer les objets de l'image par rapport à leur niveau de gris. En effet, si l'histogramme montre clairement deux modes (c'est-à-dire deux "bosses"), on peut définir un seuil T entre ces deux modes, puis appliquer un seuillage sur les pixels, tel que :

- si le niveau du pixel est inférieur à T , alors le pixel est dans la classe 0 ;
- sinon le pixel est en classe 1

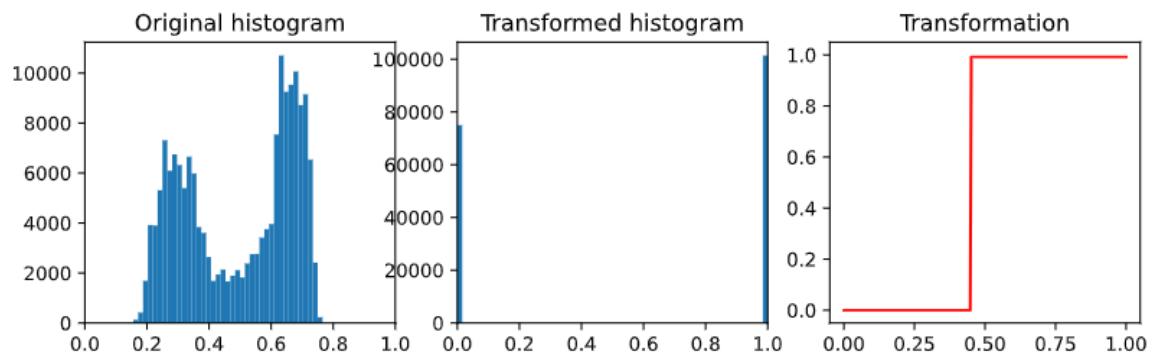


FIGURE 3 – Threshold avec un threshold posé à 0.45

Poser un tel seuil permet d'obtenir une image binaire dont les pixels n'ont que deux valeurs. Il existe plusieurs méthodes qui calculent automatiquement le seuil, comme la méthode d'Otsu que nous avons aussi tester pour la première soutenance.

La transformation par histogramme :

Une transformation d'histogramme consiste à appliquer une fonction mathématique à la distribution d'intensité. En général, les transformations sont utiles pour améliorer la qualité visuelle d'une image, mais sont rarement nécessaires dans un traitement automatique.

La transformation, notée T, est appliquée sur les intensités des pixels pour modifier leurs valeurs :

$$j = T(i)$$

où j et i sont respectivement les intensités de la nouvelle image et de l'image originale. En conséquence, l'histogramme de la nouvelle image diffère de l'histogramme de l'image originale.

Filtre de Sobel :

Nous utilisons également le filtre de sobel qui est un calcul du gradient de l'intensité de chaque pixel pour indiquer les variations du clair au sombre pour la détection des contours. Il est donc aussi utilisé pour la détection des contours d'une image. Il s'agit d'un des opérateurs les plus simples qui donne toutefois des résultats corrects.

Pour faire simple, l'opérateur calcule le gradient de l'intensité de chaque pixel. Ceci indique la direction de la plus forte variation du clair au sombre, ainsi que le taux de changement dans cette direction. On connaît alors les points de changement soudain de luminosité, correspondant probablement à des

bords, ainsi que l'orientation de ces bords.

En termes mathématiques, le gradient d'une fonction de deux variables (ici l'intensité en fonction des coordonnées de l'image) est un vecteur de dimension 2 dont les coordonnées sont les dérivées selon les directions horizontale et verticale. En chaque point, le gradient pointe dans la direction du plus fort changement d'intensité, et sa longueur représente le taux de variation dans cette direction. Le gradient dans une zone d'intensité constante est donc nul. Au niveau d'un contour, le gradient traverse le contour, des intensités les plus sombres aux intensités les plus claires.

Le filtre de Sobel calcule une approximation assez inexacte du gradient d'intensité, mais cela suffit en pratique dans beaucoup de cas. En effet, il n'utilise qu'un voisinage (généralement de taille 3×3) autour de chaque point pour calculer le gradient, et les poids utilisés pour le calcul du gradient sont entiers.

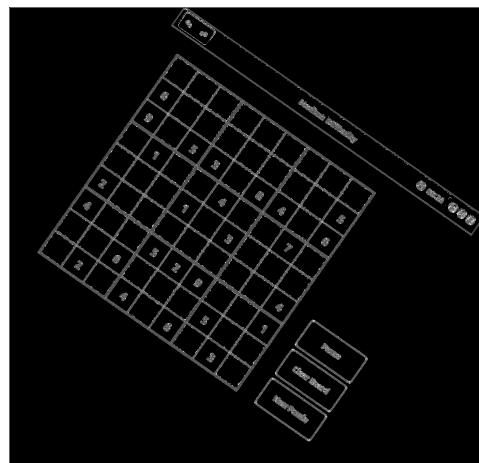


FIGURE 4 – Démonstration de sobel

L'égalisation d'histogramme :

$$T(i) = \frac{1}{MN} \sum_{k=0}^i (n_k)$$

où M et N sont la taille de l'image et n_k est le nombre de pixels d'intensité k. Cette transformation a pour but de répartir l'histogramme sur toute la gamme d'intensité, et de rendre l'histogramme aussi plat que possible. La conséquence est une augmentation du contraste de l'image. Il s'agit d'une méthode entièrement automatique qui ne nécessite aucun réglage de paramètres.

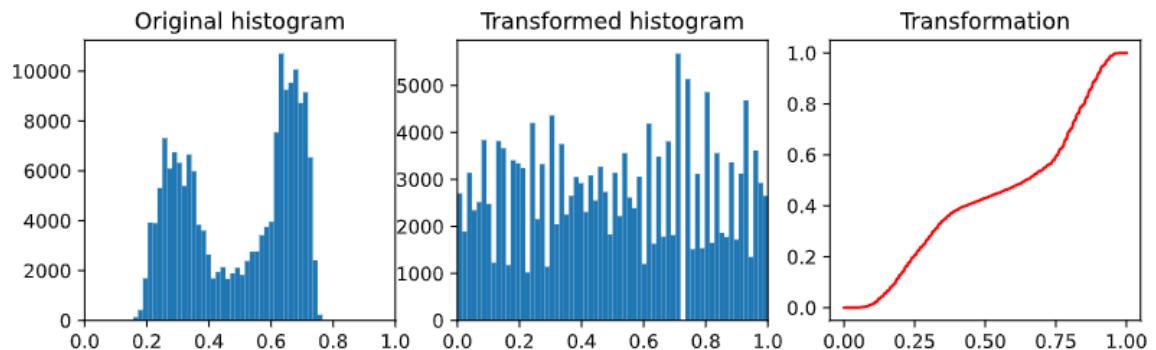


FIGURE 5 – Démonstration de l’application de l’égalisation d’histogramme

L'égalisation de l'histogramme est un autre moyen d'améliorer le contraste et tend à rendre les détails plus visibles.

Filtre Gaussien :

La première méthode, consiste à appliquer une matrice de convolution $3 * 3$, sur chaque pixel de l'image. Chaque composante de couleur du pixel est modifiée en fonction du filtre utilisé et des pixels environnants. Cette méthode se base sur le fait que les pixels d'une image sont en interaction les uns avec les autres. On applique donc la matrice sur le pixel, en sommant les produits des composantes colorées des pixels voisins avec la valeur de la matrice correspondante. On divise ensuite le résultat par la somme des valeurs de la matrice. Voici la matrice de convolution généralement utilisé pour ce filtre :

$$\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

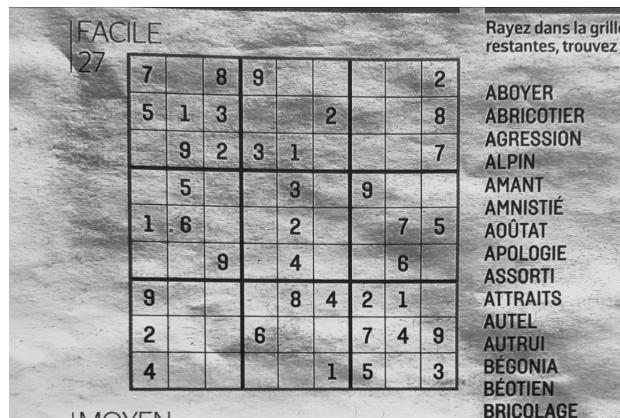


FIGURE 6 – Application du filtre de Gauss

Filtre Médian :

Cette méthode consiste à appliquer une matrice de convolution $3 * 3$, sur chaque pixel de l'image, comme le précédent. A la différence qu'on utilise la matrice suivante :

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

On remplace ensuite la valeur du pixel par la valeur médiane des pixels voisins, après application du masque. Le principe de ce modèle est le suivant : On parcourt l'image pixel par pixel, en appliquant le masque sur chaque pixel et sur ses voisins. On applique la formule suivante sur chacun des pixels voisins :

$$\text{Luminosité} = 0.3*r + 0.59*g + 0.11*b$$

Une fois le masque parcouru, on classe les valeurs du tableau, et on prend la valeur médiane. On remplace ensuite les composantes du pixel par celles du pixel correspondant.

5 Segmentation et rotation

5.1 Préparation de la segmentation

Pour cette partie, le Sobel ne suffit point. Il faut donc ajouter un autre type de filtre convolutionnel comme par exemple le Canny suivi d'un Hystérésis. Cela va nous permettre d'obtenir une image dont la grille de sudoku sera plus épaisse et donc plus facile pour l'algorithme de la détecter.

La première étape du filtre de Canny est de réduire le bruit de l'image originale avant d'en détecter les contours. Ceci permet d'éliminer les pixels isolés qui pourraient induire de fortes réponses lors du calcul du gradient, conduisant ainsi à de faux positifs.

Un filtrage gaussien en deux dimensions est utilisé. Voici l'opérateur de convolution :

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Après le filtrage, l'étape suivante est d'appliquer un gradient qui retourne l'intensité des contours. L'opérateur utilisé permet de calculer le gradient suivant les directions X et Y, il est composé de deux masques de convolution, un de dimension 3×1 et l'autre 1×3 :

$$G_x = [-1 \ 0 \ 1]; G_y = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

La valeur du gradient en un point est approximée par la formule :

$$|G| = |G_x| + |G_y|$$

et la valeur exacte est :

$$|G| = \sqrt{G_x^2 + G_y^2}$$

Les orientations des contours sont déterminées par la formule :

$$\theta = \pm \arctan \left(\frac{G_y}{G_x} \right)$$

Nous obtenons finalement une carte des gradients d'intensité en chaque point de l'image accompagnée des directions des contours.

La carte des gradients obtenue précédemment fournit une intensité en chaque point de l'image. Une forte intensité indique une forte probabilité de présence d'un contour.

Toutefois, cette intensité ne suffit pas à décider si un point correspond à un contour ou non. Seuls les points correspondant à des maxima locaux sont considérés comme correspondant à des contours et sont conservés pour la prochaine étape de la détection. Un maximum local est présent sur les extrema du gradient, c'est-à-dire là où sa dérivée selon les lignes de champs du gradient s'annule.

La différenciation des contours sur la carte générée se fait par seuillage à hystérésis. Cela nécessite deux seuils, un haut et un bas ; qui seront comparés à l'intensité du gradient de chaque point. Le critère de décision est le suivant.

Pour chaque point, si l'intensité de son gradient est :

- Inférieur au seuil bas, le point est rejeté ;
- Supérieur au seuil haut, le point est accepté comme formant un contour ;
- Entre le seuil bas et le seuil haut, le point est accepté s'il est connecté à un point déjà accepté.

Une fois ceci réalisé, l'image obtenue est binaire avec d'un côté les pixels appartenant aux contours et les autres. Ainsi, la détection des lignes et des colonnes et la rotation seront plus facile.

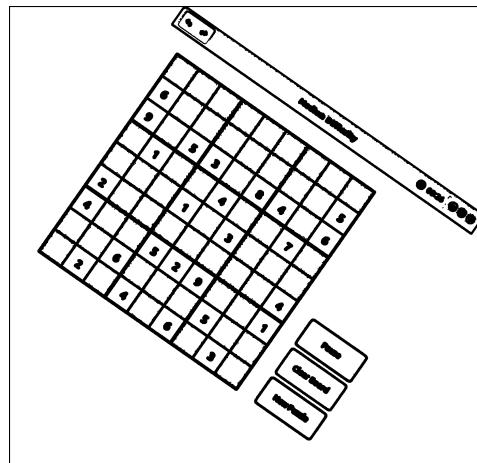


FIGURE 7 – Démonstration du filtre hystérésis

5.2 Détection de lignes et colonnes

Cela représente la plus difficile de la partie de la rotation de l'image. Pour déterminer l'angle de la rotation, nous avons utilisé le procédé de la transformée de Hough. C'est une technique qui permet de reconnaître des formes dans le traitement d'images numériques. Ici, nous l'utiliserons pour détecter les lignes et colonnes de notre sudoku.

La transformée de Hough consiste à représenter chaque point de contour détecté dans un espace de paramètres à deux dimensions :

- une droite est caractérisée par deux paramètres, elle est donc représentée par un point dans cet espace de paramètres ;
- si l'on considère l'ensemble des droites passant par un point, l'image de cet ensemble est une courbe dans l'espace de paramètres ;

La transformée de Hough d'un point de l'image analysée est la courbe de l'espace des paramètres correspondant à l'ensemble des droites passant par ce point. Si des points sont colinéaires, alors toutes les courbes de l'espace de paramètres se coupent au point représentant la droite en question.

Une droite (D) peut être caractérisée par des paramètres polaires (ρ, θ) :

- ρ est la distance de la droite à l'origine du repère ;
- θ est l'angle que fait la perpendiculaire à la droite avec l'axe x ;

Les coordonnées (x, y) des points de cette droite vérifient l'équation dite « normale » :

$$\rho = x \cdot \cos(\theta) + y \cdot \sin(\theta)$$

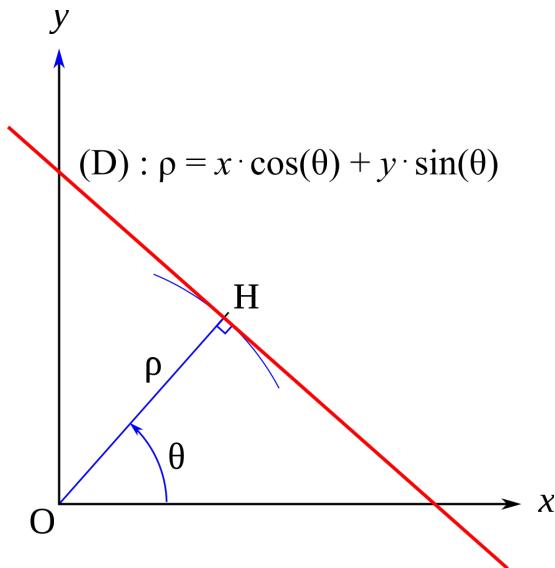


FIGURE 8 – Équation cartésienne d'une droite sous sa forme "normale"

On suppose tout d'abord que si des lignes ou des segments de droites sont présents dans une image, ils feront partie des contours présents dans l'image. On commence donc dans un premier temps par identifier tous les points de contours de cette image, notamment grâce au filtre de Canny et d'hystérosis.

Chacun des points des contours ainsi identifiés (x, y) va alors permettre une projection dans un plan (le plan transformé) des coordonnées polaires de toutes les droites passant par ce point. Les équations des droites passant en chacun de ces points (x, y) sont représentées grâce à l'équation cartésienne de la droite sous sa forme "normale". Au point (x, y) du contour, on fait donc correspondre une courbe : $\rho = f(\theta)$ où θ prend toutes les valeurs possibles de 0 à 2π rad (0 à 360°), avec :

$$f(\theta) = x * \cos(\theta) + y * \sin(\theta)$$

Ainsi pour faire simple, dans la transformée de Hough chaque ligne est un vecteur de coordonnées paramétriques. En transformant toutes les lignes possibles qui passent par un point, c'est-à-dire en calculant la valeur de ρ pour chaque θ , on obtient une sinusoïde unique appelée « espace de Hough ». Si les courbes associées à deux points se coupent, l'endroit où elles se coupent dans l'espace de Hough correspond aux paramètres d'une droite qui relie ces deux points.

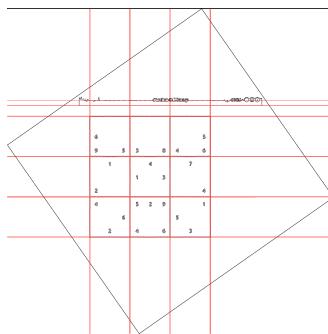


FIGURE 9 – Démonstration de la transformée de Hough

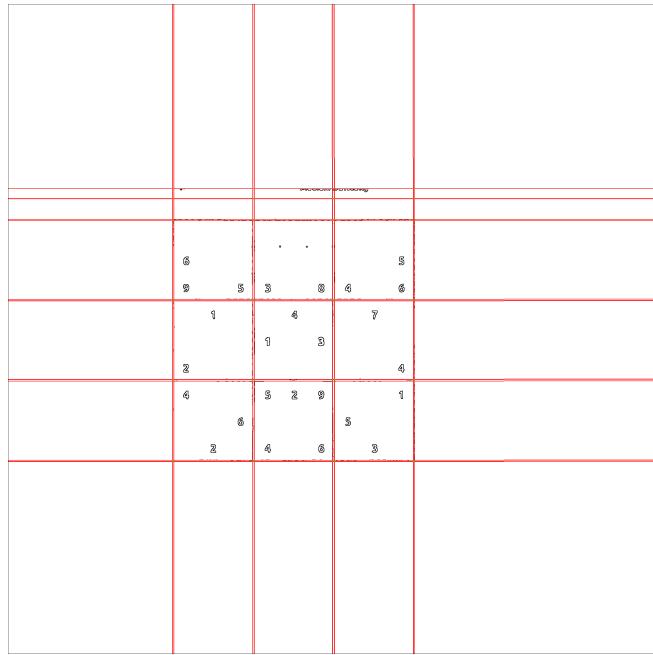


FIGURE 10 – Démonstration de la transformée de Hough avec les contours détournés de l'image originale

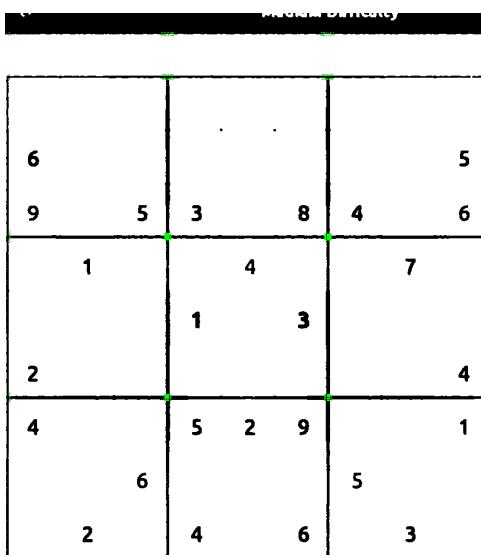


FIGURE 11 – La grille de sudoku uniquement est gardé sans les contours

5.3 Détection de l'angle

Ici aussi pour la détection de l'angle, nous utilisons l'algorithme de Hough. A l'aide de l'algorithme de Hough, nous pouvons néanmoins connaître l'angle theta qui apparaît le plus souvent sur un image. Nous en déduisons aisément que cette occurrence fréquente d'un angle theta permet de distinguer d'autres angles. Donc nous en connaissons l'angle de la rotation de notre sudoku.

Pour détecter l'angle, nous continuons la transformée de Hough que nous avons commencer dans la partie précédente qui est la détection des lignes et colonnes. La courbe obtenue avec la fonction : $f(\theta) = x^*\cos(\theta) + y^*\sin(\theta)$ est une sinusoïde.

Dans la pratique, on fait varier θ de 0 à π rad (0 à 180°) et on considère un rayon algébrique (positif ou négatif). Le paramètre ρ va prendre des valeurs entre ${}^\circ R$ et $+R$ où R est la grande diagonale de l'image : $\sqrt{R} = \sqrt{l^2 + h^2}$, l étant la largeur de l'image et h sa hauteur.

Ainsi, lorsque les courbes $\rho(\theta)$ correspondant à deux points se rencontrent, le point d'intersection caractérise la droite passant par les deux points. Donc, dans le plan (ρ, θ) , on associe à chaque point une densité correspondant au nombre de courbe qui y passent. Plus la densité est importante, plus il y a des points de contour situés sur cette droite. La droite est donc très probablement un segment dans l'image. En pratique, l'espace transformé de Hough sera représenté par une image dont les abscisses seront les angles θ , dont les ordonnées les valeurs de ρ et dont l'intensité au point quelconque (ρ, θ) est le nombre d'occurrences de (θ, ρ) provenant de l'image d'origine.

Aucune hypothèse de continuité des droites ou segments de droite de l'image de départ n'est faite ici, ce qui rend la transformée robuste à l'absence de points : masquage partiel des droites, et au bruit dans l'image.

5.4 Rotation de l'image

Concernant la rotation de l'image, nous avons choisi d'utiliser la translation de Shear.

La translation de Shear consiste à incliner une image en suivant une diagonale penchée à un certain angle. Ainsi, nous appliquons une formule à chaque point de l'image afin d'obtenir ses nouvelles coordonnées selon l'angle choisi.

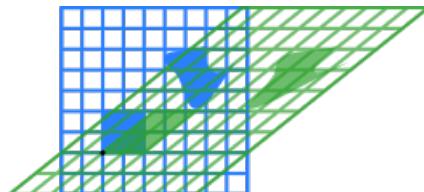


FIGURE 12 – Une transvection horizontal du plan avec le coefficient $m = 1,25$, illustré par son effet (en vert) sur une grille rectangulaire et quelques figures (en bleu). Le point noir est l'origine.

Dans le plan $R^2 = R \times R$, une transvection horizontale (ou transvection parallèle à l'axe des x) est une fonction qui amène un point générique de coordonnées (x,y) au point $(x + m * y, y)$; où m est un paramètre fixe, appelé le facteur de transvection.

L'effet de cette translation est de déplacer chaque point horizontalement d'une quantité proportionnelle à sa coordonnée y . Tout point situé au-dessus de l'axe x est déplacé vers la droite (augmentation de x) si $m > 0$, et vers la gauche si $m < 0$. Les points situés sous l'axe x se déplacent dans la direction opposée, tandis que les points situés sur l'axe restent fixes.

Les lignes droites parallèles à l'axe des x restent là où elles sont, tandis que toutes les autres lignes sont tournées, selon différents angles, autour du point où elles croisent l'axe des x . Les lignes verticales, en particulier, deviennent des points de repère. Les lignes verticales, en particulier, deviennent des lignes obliques de pente $1/m$. Par conséquent, le facteur de transvection m est la co-tangente de l'angle ϕ par lequel les lignes verticales s'inclinent, appelé angle de transvection.

Expliquons ceci dans un cas général. Pour un espace vectoriel V et un sous-espace W , une transvection fixant W translate tous les vecteurs dans une direction parallèle à W . Pour être plus précis, si V est la somme directe de W et W , et que l'on écrit les vecteurs sous la forme :

$$v = w + w$$

de manière correspondante, la transvection typique L fixant W est :

$$L(v) = (Lw + Lw) = (w + Mw) + w$$

où M est un mappage linéaire de W en W .

Par conséquent, en termes de matrice de blocs, L peut être représenté comme suit :

$$\begin{pmatrix} I & M \\ 0 & I \end{pmatrix}$$

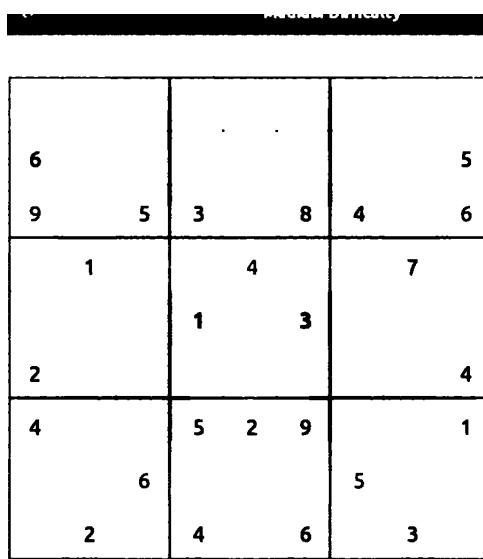


FIGURE 13 – Démonstration de la rotation sur une image avec le filtre hystérésis

5.5 Segmentation

Dans un premier temps, nous avons développé un algorithme qui nous segmente notre grille. Son concept, même intrinsèquement mathématique, reste simple à utiliser et à comprendre. Une image est parcourue verticalement et horizontalement. Des marqueurs de couleurs sont mis dessus tout en repérant les occurrences les plus répandues de séquences grille/non-grille. Or comme les cases à segmenter sont des cubes, il est simple de comprendre que toutes les cases carrées

seront détectées et segmentées.

Cependant, nous nous sommes rendus compte que cette implémentation ou ce choix n'était pas le plus optimisé.

Premièrement, pour que notre programme soit capable de détecter les différents caractères au sein du sudoku, il faut que le document soit 'parfait' ce qui justifie pleinement la phase de pré-traitement précédente.

La binarisation est obligatoire, le programme détection ne peut fonctionner que s'il détecte des pixels noirs au sein de l'image. Les différents filtres appliqués ensuite vont limiter au maximum les artefacts voire en avoir aucun grâce notamment au threshold.

La rotation est elle aussi primordiale car sans elle, il sera impossible de détecter correctement les contours de notre sudoku ce qui entraînera une nouvelle fois un problème lorsque l'on passera à l'interprétation des caractères au sein de la zone détectée.

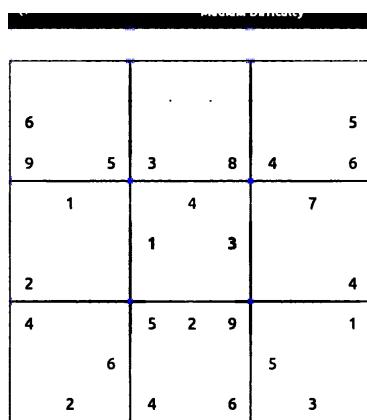


FIGURE 14 – Démonstration de la segmentation

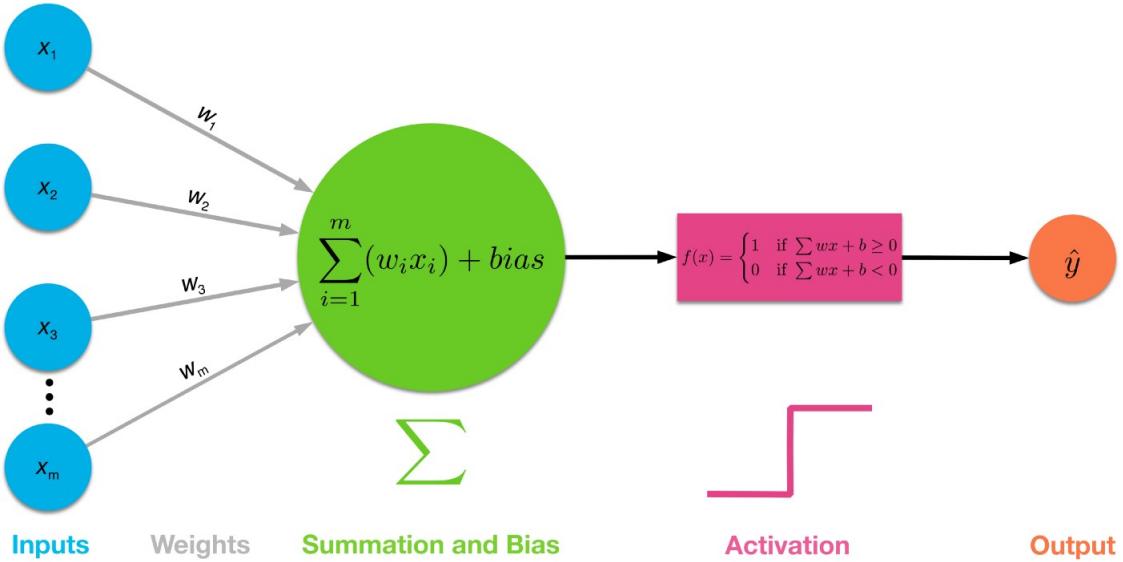
6 Réseau de neurones

6.1 Introduction

L'étape la plus importante dans la réalisation d'un OCR est le réseau de neurones car en plus d'être la partie la plus compliquée et la plus longue à réaliser, c'est l'étape pilier du programme. Elle permet de reconnaître les caractères, ce qui est le but premier d'un OCR.

6.2 Qu'est ce que c'est ?

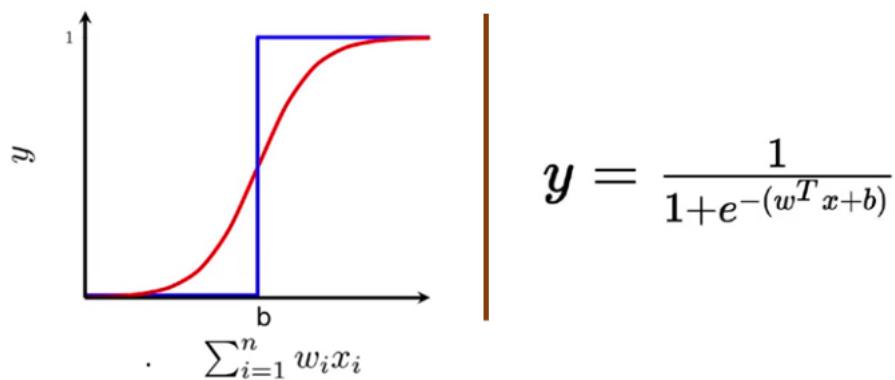
Un réseau neuronal est un ensemble de neurone, et dans le contexte une modélisation mathématique d'un neurone biologique, prenant plusieurs entrées mais ne renvoie qu'une sortie. Un neurone est entre autre relié par un synapse, le noyau recevant les informations des entrées x_1, x_2, \dots, x_n . Pour chacune de ces entrées, on va leur attribuer un poids synaptique (w_1, w_2, \dots, w_n) puis effectue une somme de ces poids. Si le nombre dépasse un certain seuil, l'information est transmise à la couche suivante, une fonction d'activation. Si le poids est inférieur au seuil, l'information n'est pas transmise.



6.3 Implémentation

L'implémentation la plus adaptée à l'exercice est l'implémentation de neurones sous forme de perceptron / neurone sigmoïde. Le modèle perceptron prend plusieurs entrées à valeur réelle et donne une seule sortie binaire. Dans le modèle de perceptron, chaque entrée x_i est associée à un poids w_i . Les poids indiquent l'importance de l'entrée dans le processus de décision. La sortie du modèle est décidée par un seuil W_0 si la somme pondérée des entrées est supérieure au seuil W_0 la sortie sera 1 sinon la sortie sera 0. En d'autres termes, le modèle se déclenche si la somme pondérée est supérieure au seuil. D'après la représentation mathématique, on peut dire que la logique de seuillage utilisée par le perceptron est très sévère.

Nous introduisons les neurones sigmoïdes dont la fonction de sortie est beaucoup plus lisse que la fonction en escalier. Dans le neurone sigmoïde, une petite variation de l'entrée ne provoque qu'une petite variation de la sortie, contrairement à la sortie en escalier. Il existe de nombreuses fonctions ayant la caractéristique d'une courbe en forme de "S", appelées fonctions sigmoïdes. La fonction la plus couramment utilisée est la fonction logistique.



Nous ne voyons plus de transition nette au seuil b . La sortie du neurone sigmoïde n'est pas 0 ou 1. Il s'agit plutôt d'une valeur réelle comprise entre 0 et 1 qui peut être interprétée comme une probabilité.

De notre côté, après avoir lu plusieurs articles sur le fonctionnement d'un réseau de neurones sigmoïds, nous avons implémenté des structures représentant un cerveau ainsi que les différentes couches et neurones qui le composent. Notre programme va dans un premier temps donner des valeurs d'entrées au réseau de neurones tout en connaissant au préalable la valeur de sortie attendue. Ainsi, on peut calculer la différence entre la réponse du programme et la réponse correcte. Cette différence peut alors être utilisée pour ajuster

progressivement les poids et biais des neurones jusqu'à ce que le cerveau soit capable de renvoyer systématiquement les sorties voulues.

6.4 Apprentissage

Le but de l'apprentissage est de permettre au réseau de neurone de s'améliorer. Cela consiste simplement à modifier les poids de chaque connexion, mais cela nécessite plusieurs étapes pour y parvenir à obtenir les nouvelles valeurs de poids. Plus on répète de fois cette opération, plus les poids des connexions devraient optimaux, en fait les poids que l'on va obtenir vont chacun converger vers leur valeur optimale. L'algorithme utilisé pour entraîner le réseau est l'Algorithme du gradient stochastique. Cela consiste à calculer la différence entre le résultat donné par le réseau de neurones et le résultat attendu. On remonte ensuite les étapes de calculs avec la dérivée de la fonction sigmoid et avec les résultats obtenus on règle les valeurs de neurones pour diminuer l'erreur jusqu'à obtenir le meilleur résultat possible.

Pour cette dernière soutenance, l'apprentissage a été complètement implémenté. Pour entraîner le réseau de neurones à lire les images, il fallait une base de données qui a été générée en python à partir des polices d'écriture de l'ordinateur. Cette base de données est essentielle pour le réseau de neurones car sans cette dernière le cerveau ne peut pas différencier les chiffres. Ce script n'est utilisé que pour générer une base de données et n'est jamais exécuté dans le fonctionnement du réseau de neurones.

6.5 La porte XOR

Nous avons réalisé pour la première soutenance un réseau de neurones qui apprend la porte XOR, le réseau que nous utilisons est un 2-1-1 avec connexions entrées-sorties c'est-à-dire qu'il existe deux neurones à la couche d'entrée et sont reliés directement à la sortie, il a une seule couche cachée contenant un neurone, et enfin un neurone en couche de sortie. On utilise aussi deux neurones de biais (un connecté au neurone caché et un connecté à la sortie) pour que l'apprentissage fonctionne mieux. On peut observer que sur les premiers tests le réseau peut se tromper, mais plus on a effectué la rétro-propagation, plus les poids seront juste et on aura une vrai porte XOR.

Cependant, pour la dernière soutenance, nous n'en avons plus l'utilité car nous avons un réseau de neurones complètement fonctionnel et entraîné pour le traitement d'image.

7 Interface graphique

7.1 Librairie GTK

GTK+ (The GIMP Tool Kit) est une librairie permettant de créer des interfaces graphiques conçue à l'origine pour le projet GIMP (The GNU Image Manipulation Program) et aujourd'hui étendu à tout type d'application et notamment celui du projet GNU : GNOME Desktop que l'on connaît tous sous le nom "d'interface graphique" des systèmes Unix en général.

L'avantage de cette librairie est d'une part sa gratuité (diffusée en open source sous licence LGPL), et d'autre part son utilisation multi langage (C, C++, Perl, Python, Caml etc.) et multi plateforme (Windows, Linux, BSD, Beos, etc.).

7.2 Notions de base

La création d'interface graphique sous GTK+ consiste à créer et manipuler des objets graphiques de GTK+. Ces objets sont appelés « Widgets ». Un Widget est une structure proposant des fonctions et propriétés permettant la manipulation de ces objets. Bien que nous sommes dans un langage procédural, le terme « objet » est à prendre au sens littéral car GTK+ introduit la notion d'héritage. Les objets graphiques héritent des membres d'un widget de base : GtkWidget.

La gestion des événements est effectuée dans une boucle événementielle par l'interception de « signaux ». Un click sur un bouton par exemple, déclenchera un signal intercepté par la boucle événementielle qui exécutera la fonction correspondante. On appelle « fonction callback » une fonction associée à un signal.

7.3 Les box

Ces derniers représentent la logique de Gtk, ce sont les éléments principaux de l'interface graphique au niveau de sa conception car ils ne sont pas vraiment visible une fois cette dernière terminée. Néanmoins cette logique a ses limites car l'imbrication d'éléments les uns dans les autres nécessite beaucoup d'appels et de très bien penser son interface avant même de l'avoir commencé. Ces éléments primordiaux dans le travail de l'interface se décomposent en deux sous-catégories, les hbox et les vbox. Les premières permettent d'empiler les éléments de manière horizontale et les secondes servent à les empiler de manière verticales. Une troisième catégorie un peu plus spécifique existe, il s'agit des Gtktables, ces dernières permettent de créer des box dans un tableau dont on peut déterminer la largeur et la hauteur et ainsi de créer une interface un peu plus équilibrée. Une fois la logique de ces éléments comprise, le travail consiste surtout à savoir où placer les éléments et à déterminer leurs actions dans l'interface.

7.4 Les widget

Ces éléments constituent le coeur graphique de l'interface. Il existe différents types de widgets, ceux destinés à afficher quelque chose comme une image ou du texte, ceux censés déterminer une action immédiate tels les boutons (contenant généralement une icône) et ceux contenant d'autres widgets à actions immédiates tels que les barres d'outils. Un autre type de widget un peu spécial existe aussi, il s'agit des barres, telles les barres de progression ou les barres de scroll pour se déplacer dans un document par exemple.

7.5 Les onglets

Ces derniers permettent d'avoir plusieurs fichiers textes ouverts à la fois et de travailler simultanément dessus, ils permettent aussi d'exécuter plusieurs fois l'ocr sur des images différentes et d'en conserver le contenu de manière immédiate ainsi que de pouvoir retourner sur un contenu précédent et le modifier sans avoir à fermer celui sur lequel l'utilisateur est actuellement. Plus tard les onglets seront connectés directement aux images auxquels ils appartiennent. En effet la structure est déjà prête et lorsqu'on cliquera sur un onglet, l'image correspondante s'affichera directement, ce qui permettra à l'utilisateur d'être plus détendu dans l'utilisation de son logiciel.

7.6 Caractérisation de notre interface graphique

Pour pouvoir faire interagir plus aisément les fonctionnalités de notre OCR, nous avons créé une interface graphique en utilisant GTK/Glade dans ce but. Comme exigé dans le cahier des charges, l'interface comporte deux fenêtres, une qui affiche l'image originale chargée depuis l'ordinateur qui restera telle quel même après traitement (binarisation, rotation, segmentation, extraction des caractères, reconnaissance des caractères). Une seconde, sous la forme d'une barre de sélection afin de choisir l'opération souhaiter sur notre image (charger l'image, la sauvegarder, résoudre le sudoku avec l'application du code et quitter l'interface graphique). Chaque image en entrée sera redimensionnée en fonction de la taille de la fenêtre de notre interface graphique.

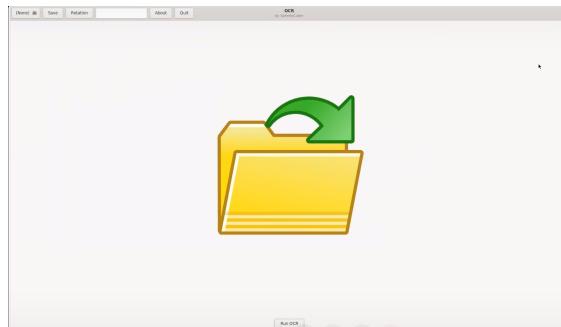


FIGURE 15 – Page d'entrée de l'interface utilisateur

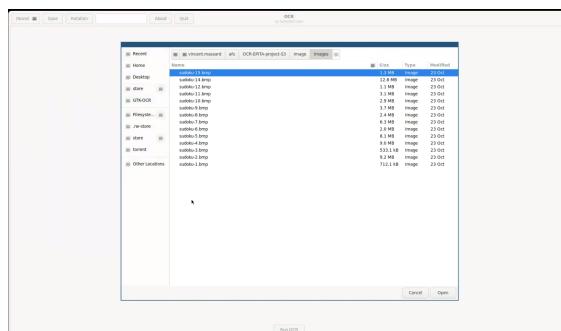


FIGURE 16 – Choix de l'image

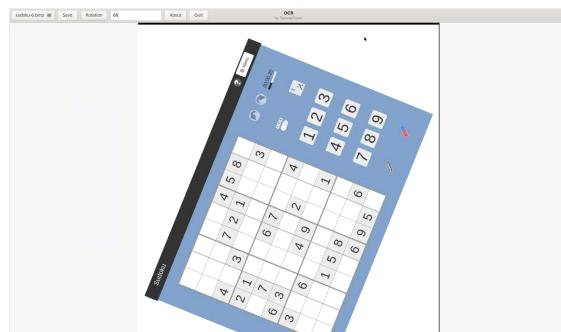


FIGURE 17 – Démonstration de l'image tournée

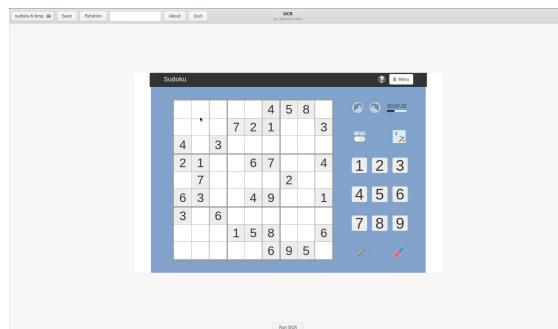


FIGURE 18 – Démonstration de la rotation fonctionnelle

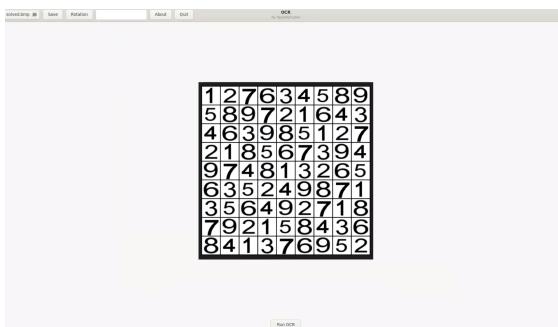


FIGURE 19 – Démonstration du résultat final de notre OCR

8 Libération de la mémoire et séparation des fichiers

Pour la dernière soutenance, nous avons décidé de rendre notre code le plus propre possible, en séparant les fichiers contenant trop de fonctions en plusieurs fichiers dépendant les uns des autres.

Quant à la libération de mémoire, elle s'avérait nécessaire, car sans cela nous ne pouvions plus faire fonctionner notre OCR. La mémoire vive se remplissait trop rapidement et l'ordinateur buguait. De plus la mémoire disponible sur les machines de l'école étant très réstrante, nous avons donc libéré tout ce qui nous était possible, comme la matrice de pixels en C, dont nous ne nous servions plus une fois le prétraitement effectué, ou encore les différents vecteurs utilisés dans notre loader d'images.

9 Site internet

9.1 La réflexion

La visibilité du projet est importante, c'est pourquoi nous avons décidé de créer un site internet pour le projet qui nous permettra d'avoir de la visibilité sur internet, importante à l'heure actuelle.

Au commencement, nous pensions utiliser des plateformes simples avec des templates déjà fait telles que « Wix » ou « WordPress ». Cependant, nous ne partions plus sur un projet créé de A à Z par notre équipe. Notre regard s'est donc tourné vers le logiciel Bootstrap Studio , dont nous avions appris l'existence au travers de vidéos liées au développement d'un site web. Des templates et du code sont déjà présent sont disponible à l'utilisation, mais ce logiciel n'étant pas facile et peu intuitif, nous avons vite choisi une autre alternative.

9.2 L'implémentation

Finalement, nous avons choisi d'apprendre les langages HTML et CSS pour créer le site web correspondant réellement à l'univers de notre projet. À l'aide de vidéos explicatives sur les langages et sur la structure d'un site web, nous avons réussi à avancer cette tâche plus vite que prévu. Ces deux langages sont les principaux dans le domaine du Web. Le HTML est utilisé pour écrire le contenu brut et le CSS l'accompagne pour le visuel (couleurs, mises en pages). Ils fonctionnent à l'aide de balises contenues dans un lexique.

Pour plus de clarté sur le site, une barre de navigation était préférable ainsi qu'une séparation des pages selon le sujet, d'où le choix d'un menu, appelé "Hamburger" dans le domaine du web. Cette tâche s'est réalisé plus rapidement que prévue. La difficulté s'est trouvée sur la réalisation du visuel. En effet, pour mettre des images en avant ou en arrière selon la page que l'utilisateur regarde, il fallait bien comprendre les indices de profondeur. De plus, pour créer des animations selon la position de la souris de l'utilisateur, une longue étude a été réalisée sur Internet.

Le site internet contient alors 4 parties, toutes reliées par le menu hamburger. Une page d'accueil, d'actualités, une sur l'équipe et une dernière contenant les différentes photos du projet. La page actualités contient l'historique des soutenances accompagné des points forts comme des points faibles et le cahier des charges avec un moyen de télécharger l'ensemble des rapports.

Après m'être documentée pour savoir à peu près comment faire une galerie photo, j'étais attirée par certaines façons mais il fallait passer dans un autre langage tel que le JavaScript ou s'aider du jQuery. Le JavaScript est un langage web qui est principalement employé dans les pages web interactives et est donc devenu aujourd'hui une partie essentielle des applications web tandis que le jQuery est une bibliothèque JavaScript libre permettant de faciliter l'écriture de scripts dans le code HTML. J'ai pris le choix de rester sur du contenu créé uniquement à l'aide du HTML et CSS.

Le site internet est hébergé sur GitHub qui est un service web d'hébergement et de gestion de développement de logiciels, utilisant le logiciel de gestion de versions Git.

10 Conclusion

Cette première partie de notre projet a été riche en expérience, en effet la chronologie des cours et du suivi du projet nous ont pris au dépourvu mais nous avons tenu à rattraper le train en direction d'une soutenance parfaite. Même si elles peuvent encore être améliorées, nos fonctions de prétraitement de l'image sont bien avancées, une segmentation correcte et un réseau neuronal efficace sous Porte XOR. Ce projet nous a appris beaucoup sur le langage C mais aussi sur les réseaux de neurones ou bien encore sur les algorithmes de détection de caractères.

Pour cette dernière soutenance, nous sommes fiers de vous montrer notre Optical Character Recognition fini.

Notre OCR est visionnable sur une interface graphique. L'utilisateur peut alors sauvegarder l'image, la tourner, observer la résolution du sudoku grâce au réseau de neurones et enfin quitter la page correctement.

Nous avons un site internet fait par nous-même sur lequel est contenu toutes les informations concernant l'historique de notre avancée au fil des soutenances, une galerie photo regroupant tous les résultats des filtres utilisés ainsi que la description de notre équipe.

Table des figures

1	Démonstration du greyscale	10
2	Démonstration de la binarisation	12
3	Threshold avec un threshold posé à 0.45	14
4	Démonstration de sobel	16
5	Démonstration de l'application de l'égalisation d'histogramme	17
6	Application du filtre de Gauss	18
7	Démonstration du filtre hystérésis	22
8	Equation cartésienne d'une droite sous sa forme "normale"	24
9	Démonstration de la transformée de Hough	25
10	Démonstration de la transformée de Hough avec les contours détournés de l'image originale	26
11	La grille de sudoku uniquement est gardé sans les contours	26
12	Une transvection horizontal du plan avec le coefficient $m = 1,25$, illustré par son effet (en vert) sur une grille rectangulaire et quelques figures (en bleu). Le point noir est l'origine.	28
13	Démonstration de la rotation sur une image avec le filtre hystérésis	30
14	Démonstration de la segmentation	31
15	Page d'entrée de l'interface utilisateur	40
16	Choix de l'image	40
17	Démonstration de l'image tournée	40
18	Démonstration de la rotation fonctionnelle	41
19	Démonstration du résultat final de notre OCR . .	41