

# Projet OCR

## Rapport de soutenance n°1

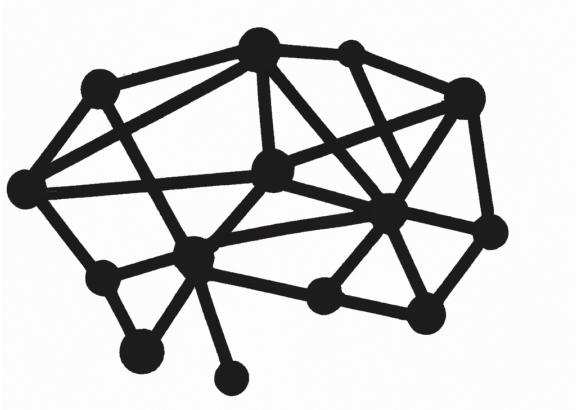
Laure MONTREDON

Irène LIN

Ambroise DURST

Florian FOGLIANI

Octobre 2023



## Sommaire

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Présentation de l'équipe et répartition des tâches</b>	<b>3</b>
<b>3</b>	<b>Aspect technique</b>	<b>5</b>
3.1	Le pré-traitement . . . . .	5
3.1.1	Conversion en noir et blanc . . . . .	5
3.1.2	Rotation de l'image . . . . .	6
3.2	Détection et découpage de l'image . . . . .	6
3.2.1	Détection des lignes . . . . .	6
3.2.2	Découpage des cases . . . . .	8
3.3	Réseaux de neurones . . . . .	9
3.3.1	Principe générale . . . . .	9
3.3.2	Notions mathématiques . . . . .	9
3.3.3	Algorithme . . . . .	10
3.4	Solver . . . . .	10
3.4.1	Liste chaînée . . . . .	11
3.4.2	Backtraking . . . . .	11
3.4.3	Formatage . . . . .	12
<b>4</b>	<b>Retard(s)</b>	<b>13</b>
4.1	Pré-traitement . . . . .	13
<b>5</b>	<b>Conclusion</b>	<b>16</b>

## 1 Introduction

L'objectif de ce projet de réaliser un logiciel de type OCR (Optical Character Recognition) capable de résoudre une grille de sudoku. L'application sera en mesure de prendre en entrée une image représentant une grille de sudoku, de la traiter et ensuite d'afficher en sortie la grille résolue.

Dans sa version définitive, cette application devra proposer une interface graphique permettant de charger une image dans un format standard, de la visualiser, de corriger certains de ses défauts, et enfin d'afficher la grille complètement remplie et résolue. La grille résolue devra également pouvoir être sauvegardée.

Pour cette première soutenance les objectifs étaient :

- le chargement d'une image et la suppression des couleurs ;
- la rotation manuelle de l'image ;
- la détection de la grille et de la position des cases ;
- le découpage de l'image (sauvegarde de chaque case sous la forme d'une image) ;
- l'implémentation de l'algorithme de résolution d'un sudoku.
- de fournir une preuve de concept de notre réseau de neurones. Pour cette preuve, il est demandé de réaliser un mini réseau capable d'apprendre la fonction OU EXCLUSIF.

## 2 Présentation de l'équipe et répartition des tâches

OCRnest est une équipe composée de 4 membres :

### Irène

Dans le projet je me charge de tout ce qui touche au pré-traitement, c'est-à-dire, la rotation, la suppression des bruits parasites, des couleurs, et le renforcement de contrastes.

### **Ambroise** : Chef de projet

C'est avec plaisir que je m'occupe de la partie "traitement" du projet. J'ai dû me renseigner sur les méthodes de base du machine learning, à savoir les réseaux de neurones et l'algorithme de descente de gradient. Le sujet est encore plus intéressant que ce que je pensais, bien qu'assez rude en mathématiques (c'est la première fois qu'elle me servent autant) En tout cas j'espere pouvoir mener à bien, depuis ma partie, l'ensemble du projet :)

### **Laure**

Pour cette première soutenance, je me suis occupée du solver. J'ai étudié de nombreux types d'algorithmes différents pour trouver quelle proposition était la plus optimisée.

### **Florian**

Pour ce premier rendu, mon objectif était la détection des lignes et le découpage des grilles du Sudoku. C'est la première fois que j'implémente des notions de géométrie aussi complexes mais suis heureux de découvrir le fonctionnement du traitement d'images. Je suis content de participer à un projet si intéressant, complet, et plein d'apprentissages.

### 3 Aspect technique

#### 3.1 Le pré-traitement

##### 3.1.1 Conversion en noir et blanc

Pour effectuer la conversion en noir et blanc, on parcours chaque pixel de l'image colorée. L'objectif est de remplacer les couleurs en calculant la moyenne des valeurs RGB pour chaque composante du pixel. En d'autres termes, pour chaque pixel P avec des composantes RGB (R, G, B), nous calculons les nouvelles composantes R', G', B' comme suit :

$$R' = G' = B' = \frac{1}{3}R + \frac{1}{3}G + \frac{1}{3}B$$

Figure 1: formule pour obtenir la moyenne des valeurs RGB

Cependant, pour obtenir une image en noir et blanc avec un seuil, nous devons ensuite comparer la valeur de luminance moyenne à ce seuil. Ici, le seuil est de 130, ce qui signifie que si la moyenne de composantes R', G', B' est supérieure à 130, le pixel devient blanc, sinon, il devient noir.

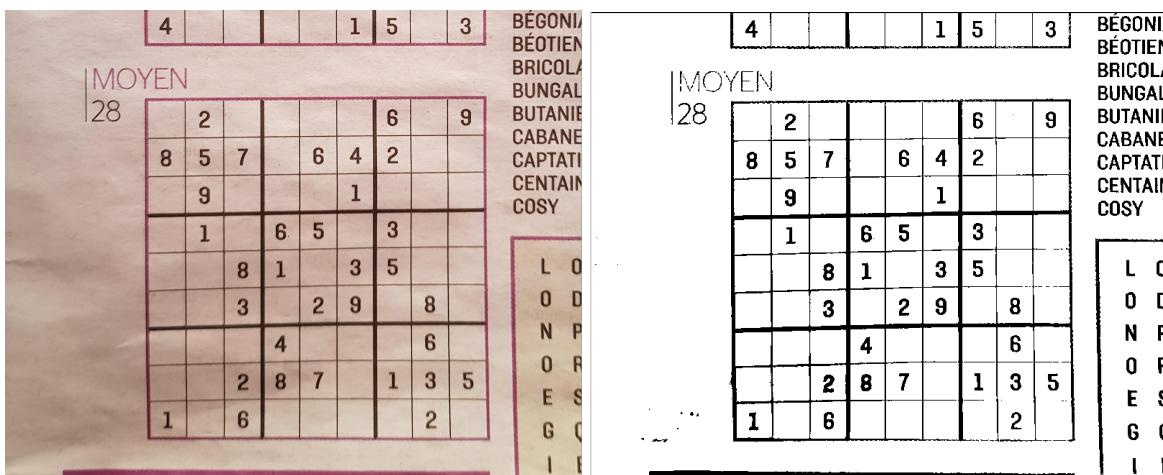


Figure 2: Avant / après conversion en noir et blanc

### 3.1.2 Rotation de l'image

Il existe diverses méthodes de rotations, mais pour notre projet, cette opération est réalisée par le SDL (Simple DirectMedia Layer) au moment du rendu, ce qui signifie que l'image d'origine n'est pas altérée directement. Au lieu de cela, SDL génère une nouvelle image orientée en fonction de l'angle de rotation spécifié, sans modifier l'image source. Cette nouvelle image est ensuite sauvegardée sous le nom de "img\_oriented.png" pour une utilisation future. Lorsque l'angle de rotation est de 0 degrés, cela signifie que l'image reste inchangée. Dans ce cas, l'image d'origine est simplement rendue à l'écran. Lorsque la valeur de l'angle de rotation est de 0, l'image reste inchangée. En revanche, lorsque l'angle de rotation est différent de 0, une rotation de l'image est effectuée selon l'angle spécifié

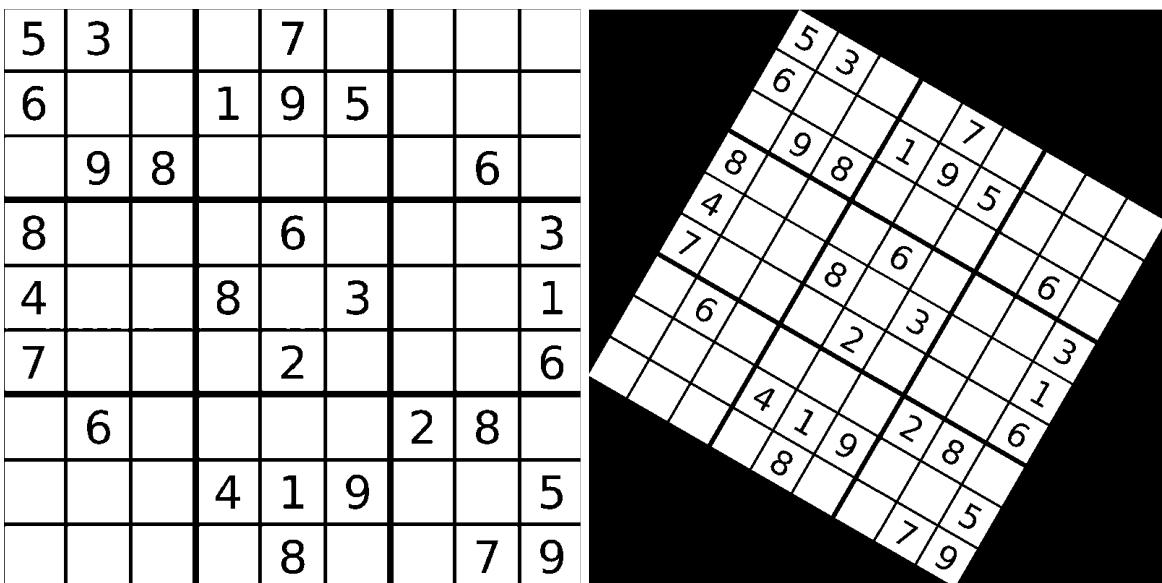


Figure 3 :Avant / après rotation de la grille de sudoku

## 3.2 Détection et découpage de l'image

### 3.2.1 Détection des lignes

Pour la détection des lignes nous avons décidé d'utiliser le transformée de Hough. Le transformée de Hough fonctionne sur le principe suivant : pour

chaque pixel de l'image, l'on va étudier l'ensemble des droites passant par celui-ci et dans une matrice de correspondance on incrémentera la case de la droite associé en coordonnée polaire. Une droite en coordonnée polaire est représenté par le couple ( $\rho$ ,  $\theta$ ) tel que  $\rho$  est la distance de la droite à l'origine du repère et  $\theta$ , l'angle que fait la perpendiculaire à la droite avec l'axe x.

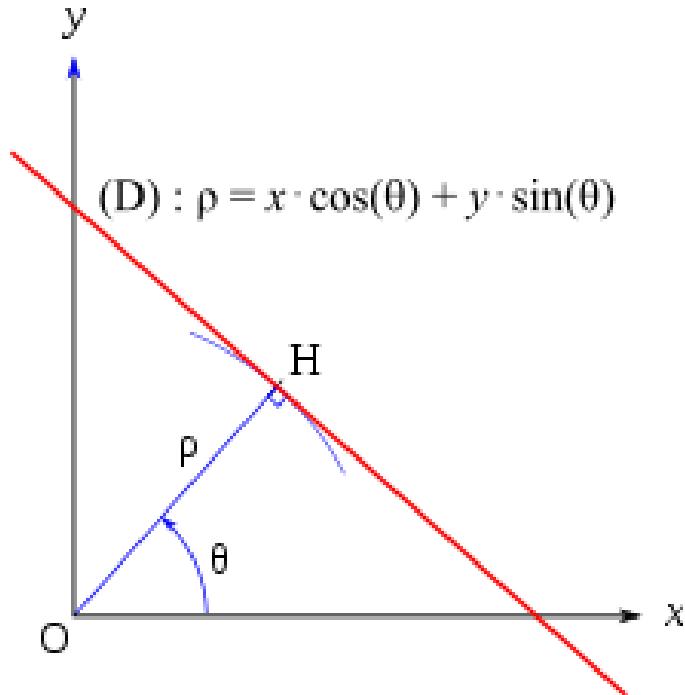


Figure 4 : Cordonnées polaires d'une droite

La matrice de correspondance représente en ligne l'intervalle de  $\rho$  allant de  $[-p, +p]$  résultat d'une formule en fonction de l'angle  $\theta$  et de la position du pixel et en colonne l'angle  $\theta$  allant de  $[-90, +90]$ . Ainsi l'on va pour chaque pixel, étudie les droites d'angles  $-90$  jusqu'à  $+90$  tout en calculant leurs  $\rho$  associé. Ce couple  $[\rho, \theta]$  représentera une case dans la matrice de correspondance.

Ensuite, nous détectons les maximums dans la matrice de correspondance. Ces maximums représentent les droites qui passent par le plus de points dans l'image. Dans le cas du sudoku cela représente la grille avec ses lignes horizontales et verticales.

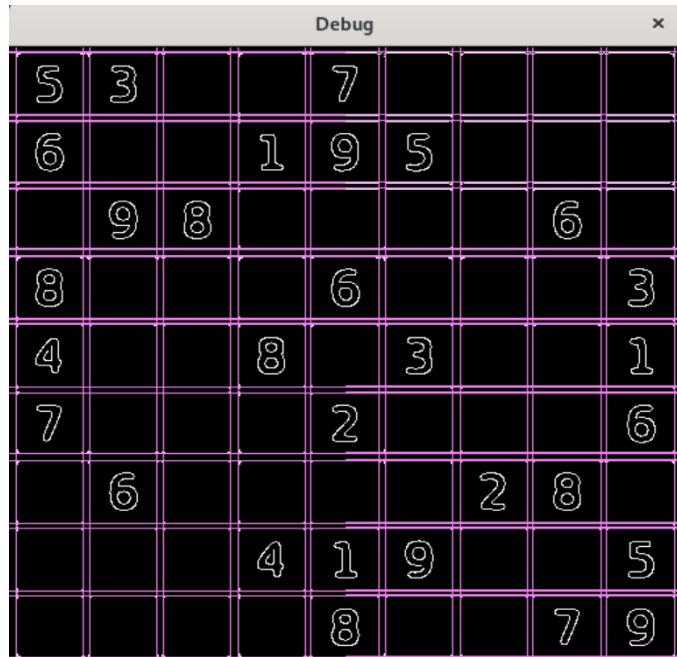


Figure 5 : Résultat de la détection des droites

### 3.2.2 Découpage des cases

Pour le découpage nous avons décidé de récupérer du transformée de Hough, l'ensemble des lignes horizontales d'un côté et l'ensemble des lignes verticales de l'autre. Ensuite, l'on cherche les 10 lignes verticales et les 10 lignes horizontales correspondantes à la grille de Sudoku, à distances égales deux à deux. Pour cela, l'on va étudier leurs distances par rapport à l'origine (leurs coordonnées rho). En premier temps, nous allons triés les lignes horizontales et verticales par ordre croissantes. Ensuite, nous déterminons les 10 lignes horizontales et 10 lignes verticales à distance égales deux à deux. Ainsi, nous obtenons la grille de Sudoku uniquement. Enfin, nous allons étudier les points d'intersections des droites entre elles pour obtenir l'ensemble des côtés de chaque case. Grâce à SDL, nous enregistrerons chaque case sous la forme d'une image grâce à leurs bords supérieurs droits et leurs bords inférieurs gauche. Le format de fichier enregistrés sont au format mat\_ligne\_colonne.

### 3.3 Réseaux de neurones

#### 3.3.1 Principe générale

Plusieurs notions ont dû être acquises pour implémenter correctement le traitement qui servira à la reconnaissance des nombres. Des méthodes mêlant algorithmie et mathématiques sont entrées en jeu, en voici donc un résumé structuré:

Nous utilisons du machine learning, cette branche de l'IA utilise une structure de donnée clef autour de laquelle toute la phase de traitement va tourner, à savoir un réseau de neurones. Ce réseau de neurones, initié par un programme, est premièrement constitué de valeurs choisies au hasard. Puis en suit une phase d'apprentissage, durant laquelle grâce à différents algorithmes, les valeurs du réseau sont modifiées, jusqu'à obtenir un réseau opérationnel. Pour finir, il ne reste plus qu'à tester le réseau pour s'assurer de son bon fonctionnement.

Ainsi, la création de notre réseau de neurones peut se résumer en 3 phases: Initialisation, Apprentissage, Vérification.

A présent, en gardant en tête ces trois phases, nous pouvons davantage entrer dans les détails, en effet pour implémenter ce système, des notions mathématiques, et des algorithmes, ont dû être intégrés.

#### 3.3.2 Notions mathématiques

Des matrices sont utilisées pour représenter le réseau de neurones, ces matrices facilitent et accélèrent les calculs, permettant de simplifier et d'optimiser le programme.

Une loi normale de distribution (ou loi gaussienne), est utilisée pour initialiser aux hasards, mais avec une certaine cohérence, les valeurs du réseau de neurones. Le calcul de dérivées partielles, nécessaire aux fonctionnement des algorithmes, a été implémenté.

### 3.3.3 Algorithme

A l'état actuel du projet, un seul algorithme est utilisé pour la phase d'apprentissage: la descente de gradient. Le principe général de la descente de gradient, consiste en le calcul d'un gradient, qui correspond à la liste des petites variations que chaque valeur du réseau doit subir pour améliorer la performance générale. Ainsi, on calcule ce gradient puis on applique ces petites variations à chaque valeurs, et répète l'opération autant de fois que nécessaire pour obtenir un réseau fonctionnel.

Enfin, plusieurs paramètres sont manipulables pour adapter l'apprentissage du réseau:

- Le nombre de neurones dans le hidden layer
- La vitesse d'apprentissage (rate)
- La nombre de cycle d'entraînements (epochs)

Pour cette première soutenance, le réseau est entraîné pour reproduire le comportement d'une porte XOR.

Cette tâche simple pour le réseau nécessite que 3 neurones dans le hidden layer, une vitesse d'apprentissage de 0.01 à 5, et 200 cycles d'entraînements

```
EPOCH 90 : 0.428583
EPOCH 91 : 0.000001
EPOCH 92 : 0.001135
EPOCH 93 : 0.000000
EPOCH 94 : 0.000072
EPOCH 95 : 0.000000
EPOCH 96 : 0.000000
EPOCH 97 : 0.000000
EPOCH 98 : 0.980223
EPOCH 99 : 0.000000
[0.000000, 0.000000] ==> 0.000000
[0.000000, 1.000000] ==> 0.999659
[1.000000, 0.000000] ==> 0.999827
[1.000000, 1.000000] ==> 0.000000
```

Figure 6 : Apprentissage du Xor

### 3.4 Solver

Le solver résout le sudoko, en alliant liste chaînée et backtracking la plupart des

résolutions prennent environs 0,0005s mais certaines grilles peuvent prendre jusqu'à 0.002s

### 3.4.1 Liste chaînée

Afin d'optimiser le programme de résolution par backtracking, nous avons implémenté notre propre système de liste chaînée. Cette liste contient le nombre de valeurs que peut prendre chacune des cases vides du sudoku et est triée par ordre croissant. De cette manière, les cases contenant le moins de possibilités sont traitées en priorité.

Pour cela nous avons crée une structure contenant la position de la case, le nombre de valeurs qu'elle contient et un pointeur vers l'élément de la liste chaînée.

Cette implémentation de la liste chaînée posséde trois fonctions: new\_elt,insert et free\_l.

-New\_elt crée un nouvel élément indépendant de la liste chaînée principale. Pour cela elle prend en paramètres : la position et le nombre de valeurs que peut prendre la case.

-Insert insert un nouvel élément dans une liste chaînée. Elle le placera par ordre croissant selon le nombre de valeurs possibles.

-Free\_l free la mémoire d'une liste chaînée passée en paramètre.

### 3.4.2 Backtraking

Le programme va remplir progressivement la grille en vérifiant que celle-ci est toujours potentiellement valide. Si le fait de placer un nombre rend la grille fausse alors on change le nombre, s'il n'y a aucune solution possible : on change le nombre de l'étape précédente, sinon : on fait un appel recursif de la fonction sur la nouvelle grille.

Afin d'augmenter la rapidité du traitement on remplit les cases selon le

nombre de possibilités qu'elle possède. Pour ce faire on créer une liste chaînée contenant le nombre de possibilités pour toutes les cases vides. En effectuant un backtraking depuis les cases avec un minimum de solutions vers les cases avec un maximum de solutions on minimise significativement la complexité temporelle.

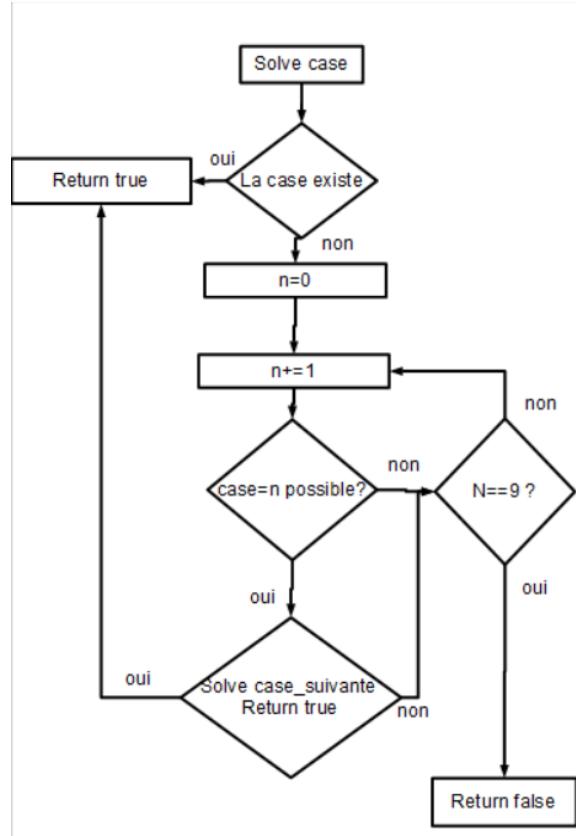


Figure 7: Algorithme du solver

### 3.4.3 Formatage

Le fichier passer en entrée est un fichier texte de type :



Figure 8: format des fichiers texte

Ce fichier est lu et est entré dans un tableau d'entiers de taille 81. Les points sont transformés en zéros pour la résolution du solveur. Ensuite, le programme enregistre le résultat dans un nouveau fichier texte avec le même format que le texte d'entrée.

## 4 Retard(s)

### 4.1 Pré-traitement

En observant le processus de conversion en noir et blanc décrit précédemment, il est évident qu'il peut y avoir des problèmes avec les images contenant du bruit, tels que des tâches ou des ombres. Par conséquent, il est judicieux d'envisager l'utilisation d'un filtre pour réduire ce bruit, ce qui est en cours de développement. La réduction de bruit fait référence à la suppression des fluctuations parasites et aléatoires de la lumière ou des couleurs dans une image, généralement présentes en raison des conditions de la prise de vue, des capteurs de caméra ou d'autre facteurs. Bien que la suppression complète du bruit soit difficile, il est possible de le réduire, au moins partiellement, pour atténuer ses effets indésirables.

Plusieurs méthodes de filtrage sont actuellement en cours de développement

pour réduire le bruit dans les images:

- Filtre de Canny : Le filtre de Canny est une technique de traitemnet d'image utilisée principalement pour détecter les contours, mais aussi pour réduire le bruit en supprimant les variations de couleur ou de luminosité qui ne sont pas associées à des contours importants. Il permet de mettre en évidence les contours tout en atténuant le bruit.
- Flou gaussien: le flou gaussien est un filtre qui répartit les poids en fonction de leur distance par rapport au pixel. Il est couramment utilisé pour flouter légèrement l'image , ce qui a pour effet de lisser le bruit. Le flou gaussien est efficace pour réduire le bruit de haute fréquence tout en préservant les caractéristiques générales de l'image.
- Histogramme: l'histogramme de l'image peut être utilisé pour améliorer le contraste et réduire le bruit. En ajustant les niveaux de luminosité ou de couleur dans l'image en fonction de leur distribution dans l'histogramme, il est possible d'atténuer certaines variations non souhaitées.

Il est courant d'utiliser une combinaison de ces méthodes pour obtenir les meilleurs résultats. La réduction de bruit est une étape importante dans le traitement d'images, car elle contribue à améliorer la qualité visuelle et permet de garantir que les informations importantes de l'image restent intactes tout en éliminant les perturbations indésirables.

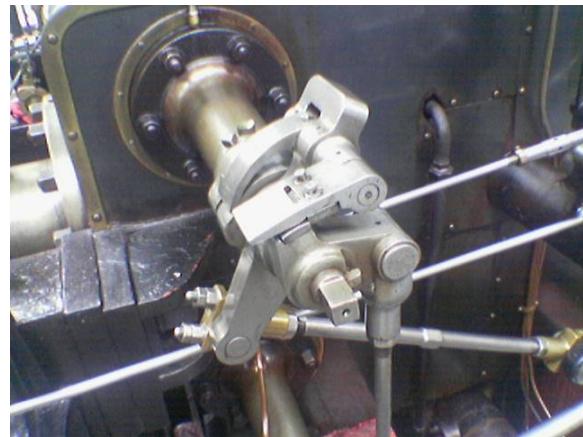


Figure 9: Image d'origine

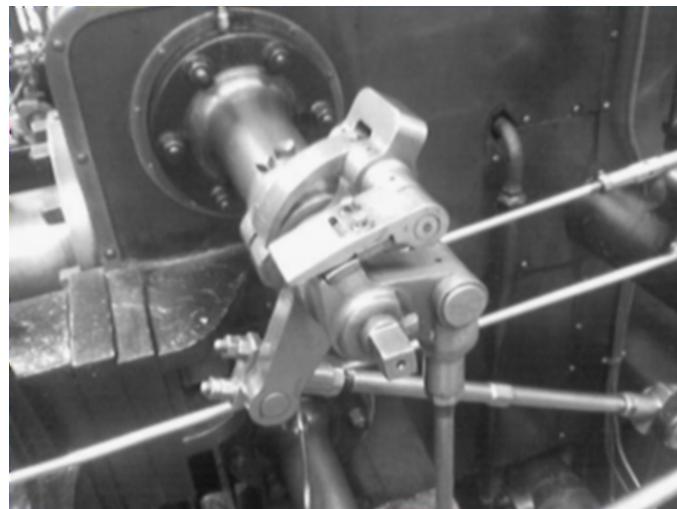


Figure 10: Image après application d'un flou gaussien

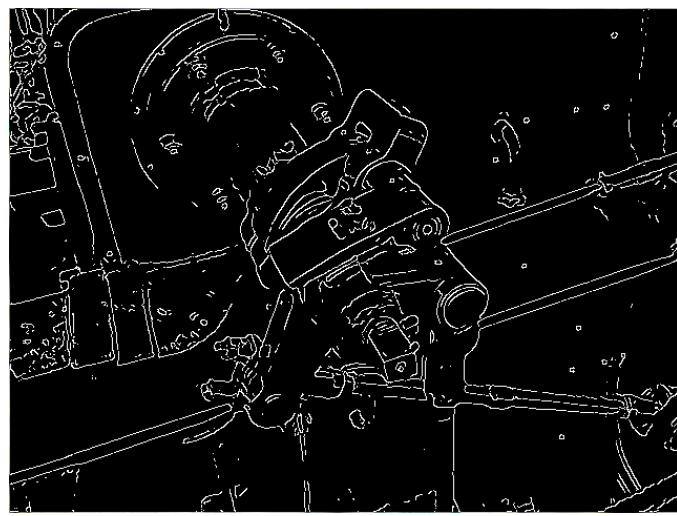


Figure 11: Filtre de Canny

## 5 Conclusion

Malgré quelques légers retards dans le prétraitement de l'image, nous sommes proches de nos objectifs. De plus, ces retards seront rapidement comblés au cours des prochaines semaines

Nous avons atteint la grande majorité de nos objectifs. Aujourd’hui, nous sommes fiers de vous présenter l’avancement de notre projet.