UNIVERSITÉ DE FRIBOURG SUISSE
UNIVERSITÄT FREIBURG SCHWEIZ

SudoClick
Reconnaissance de grilles de sudoku pour téléphones portables

Patrick Anagnostaras[1]

24 mai 2008

Department of Informatics - Master Project Report

Département d'Informatique - Departement für Informatik • Université de Fribourg - Universität Freiburg • Boulevard de Pérolles 90 • 1700 Fribourg • Switzerland

phone +41 (26) 300 84 65 fax +41 (26) 300 97 31 Diuf-secr-pe@unifr.ch
http://diuf.unifr.ch

Accepted for: Patrick Anagnostaras, SudoClick. *DIVA Group Documents Factory*, 1(1) :1–2, February 2006.

[1] patrick.anagnostaras@unifr.ch, DIVA group, DIUF, University of Fribourg

Résumé

Ce rapport présente la synthèse du travail de Master ainsi que les résultats obtenus. L'objectif principal est la reconnaissance des grilles de sudoku pour les téléphones portables. Les sujets traités portent sur le traitement des images, la binarisation des images, la détection de formes dans les images, la reconnaissance de chiffres, les réseaux de neurones et la résolution des sudokus.

Table des matières

Table des figures

1 Introduction

Les grilles de sudoku sont devenues très populaires de nos jours. Il est difficile d'ouvrir un journal ou un magazine et de ne pas y trouver des grilles de sudoku à résoudre. Il existe sur Internet des systèmes de résolution automatique mais qui nécessitent la saisie de la grille à la main. L'objectif principal de ce projet de Master est la résolution d'une grille de sudoku directement à partir d'une photo de la grille. Un second objectif est d'analyser les possibilités de résolution du sudoku directement sur téléphones portables. Ce projet est effectué avec la participation de *Swisscom Innovations*. La grande ligne de ce projet est de partir avec une image de grille de sudoku et d'en obtenir la solution. La problématique a été divisée en quatre étapes qui seront définies et discutées dans ce rapport (Fig. 1) :

- Le traitement des images prises avec les téléphones portables.
- La détection de la grille et des cases du sudoku.
- La reconnaissance des chiffres du sudoku.
- La résolution du sudoku.

Le traitement des images est nécessaire car les photos prises avec des téléphones portables sont de mauvaise qualité. Les différentes méthodes de traitement d'images testées ainsi que les résultats obtenus seront exposés dans ce rapport. Le traitement des images comprend une étape importante qui est la binarisation des images. Quatres différents algorithmes de binarisation seront comparés et testés afin de définir lequel donne la meilleure solution. Une fois les traitements et la binarisation effectués, il faut détecter la grille du sudoku. Cette détection est aussi testée avec trois différentes méthodes qui seront évaluées selon le pourcentage de détection des grilles et des cases sur les images.
Par la suite, un réseau de neurones pour la reconnaissance des chiffres dans les cases détectées dans l'étape précédente sera implémenté. Pour effectuer des tests sur le réseau de neurones, une base de données d'images sera construite pour définir deux ensembles bien distincts ; l'ensemble d'entraînement du réseau de neurones et l'ensemble de validation du réseau de neurones. Un certain nombre de paramètres sont modifiables afin d'optimiser au mieux le réseau de neurones. Les résultats de ces variations de paramètres seront exposés afin d'obtenir la meilleure reconnaissance possible des chiffres. La dernière étape expose la méthode de résolution du sudoku.

L'objectif final de ce projet est de proposer aux utilisateurs un système de résolution simple lui permettant de prendre une photo d'une grille de sudoku avec son téléphone portable et d'en obtenir la solution. Une des difficultés réside dans le fait que les appareils-photo des téléphones portables ne sont pas conçus pour prendre des photos d'objet proche de l'objectif. Il reste à soulever une question importante pour l'utilisation de ce système : faut-il proposer un système téléchargeable et fonctionnant sur les téléphones ou proposer un système de client serveur avec l'envoi des images directement depuis le téléphone en utilisant la technologie MMS ?
Nous allons essayer de répondre à ces questions tout au long de ce rapport. Pour commencer, il est important de définir les hypothèses de départ qui ont été décidées pour l'accomplissement de ce projet.

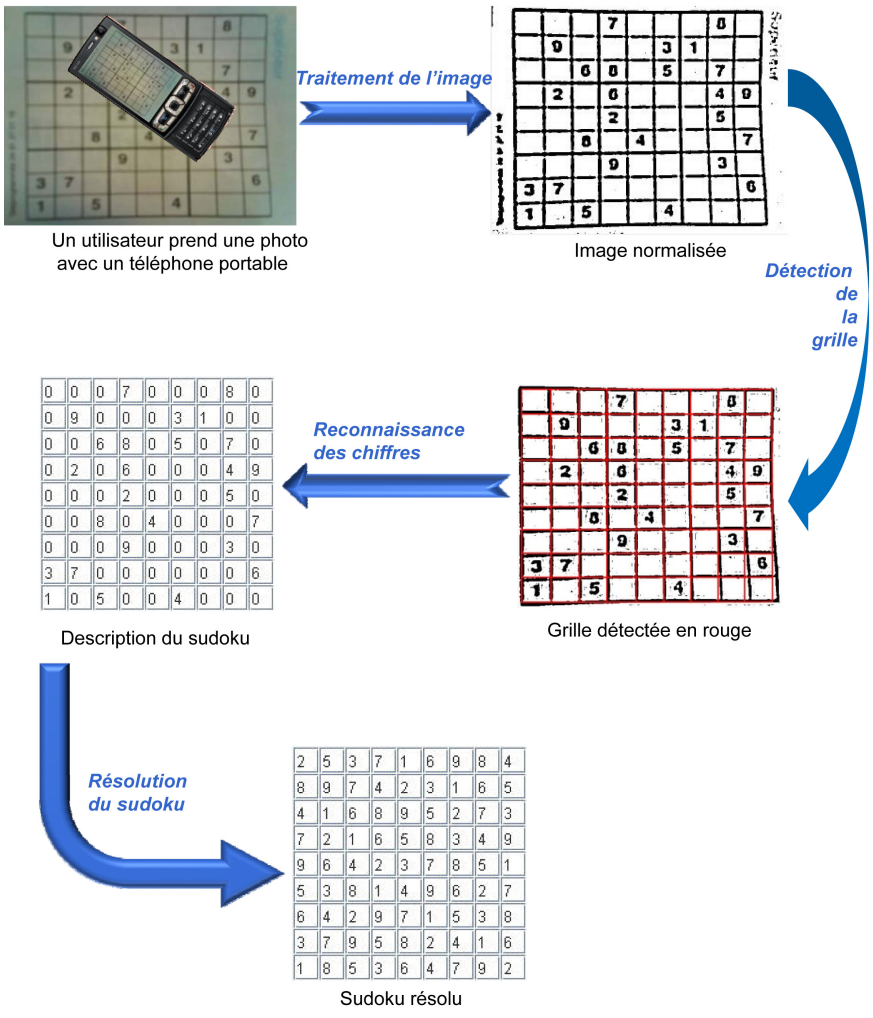

FIG. 1 – Les quatres étapes du projet

1.1 Hypothèses initiales

Les utilisateurs d'un tel système sont imprévisibles. Ils peuvent prendre des photos avec toutes sortes de singularités. Ces éléments nous imposent de définir des hypothèses de départ pour canaliser les possibilités de singularité. Les hypothèses suivantes sont faites pour ce projet :

- La première hypothèse est que les grilles de sudoku sont vierges donc pas encore remplies à la main. L'utilisateur prend une photo de la grille avant de commencer à l'annoter, cela simplifiera la reconnaissance des chiffres vu que le système n'aura pas à faire la distinction entre un chiffre manuscrit et un chiffre imprimé.

- Les téléphones actuels possèdent une résolution supérieure à 1 MegaPixel. La seconde hypothèse suppose que les photos sont prises par des téléphones portables récents ayant une résolution supérieure à 1 MegaPixel.

- Les grilles des sudokus doivent prendre la plupart de la place dans l'image avec le moins possible de bruit autour. Les images ayant des parties de grille manquante ou des grilles prises de trop loin ou de trop près avec des chiffres non reconnaissables à l'oeil nu ne seront donc pas prises en compte.

- Les images de sudoku vont certainement présenter différents types de bruits et de déformations qui devront être raisonnables dans la mesure où il est possible de voir la grille et les chiffres qui la composent.

2 Traitement des images

La numérisation de documents avec un appareil-photo comporte de nombreux avantages par rapport à un scanner. Un appareil photo est petit, facile à porter et à utiliser. Il peut être utilisé dans n'importe quelle situation et peut être utilisé pour prendre des photos de documents difficiles à scanner comme des livres, des journaux ou des inscriptions sur des immeubles, des véhicules ou d'autres objets en mouvement.

En général les systèmes basés sur des appareils-photos sont plus flexibles [5].

Il y a de nombreuses applications basées sur la reconnaissance de texte sur des images prises avec des appareils-photo, comme la reconnaissance des plaques d'immatriculations des voitures, le classement des livres, la lecture de l'identifiant des trains, la reconnaissance des panneaux de circulation, la détection de signaux de danger et la lecture de signaux dans les entrepôts. Cette méthode offre de nouvelles possibilités :

- La lecture des plaques d'immatriculation, par exemple à Londres ou dans les parkings pour permettre une facturation simplifiée à l'utilisateur.
- La détection et la traduction de symboles ; la possibilité de reconnaître du texte en utilisant des téléphones portables a un grand avenir dans des applications commerciales et militaires.
- La reconnaissance mobile pour les aveugles [15] [9] ; un système propose d'utiliser une caméra sur la tête des aveugles qui pourrait détecter et reconnaître du texte dans l'environnement de la personne et la traduire en texte to speech.
- La lecture des codes-barres dans les containers et les entrepôts de marchandise ; Lee et Kankanhalli [6] ont présenté un système utilisé dans les ports pour lire automatiquement les numéros des containers des cargos. Leur détection de texte est basée sur des verticales trouvées dans les images avec une vérification du code du container car ce dernier est standardisé dans un format fixe.

Les images résultantes de photos prises par des appareils-photos de téléphone portable souffrent de nombreux problèmes qui vont être définis dans cette section. Une fois les traitements effectués, il sera nécessaire de binariser l'image pour la suite du processus.

2.1 Description du problème

La reconnaissance des grilles de sudoku avec l'appareil-photos d'un téléphone portable est plus complexe que si les grilles étaient scannées. Avec un scanner, nous avons toute une série de constantes telles que la lumière, la résolution, la proportion, etc... Lorsqu'un utilisateur prend une photo, de nombreuses variations [3] entrent en jeu :

La lumière Les appareils-photos des téléphones portables ont beaucoup moins de contrôle sur les conditions d'éclairage des objets qu'un scanner. L'environnement entourant l'utilisateur est important lorsqu'il prend des photos [2]. La lumière peut varier à cause de l'environnement physique, à cause de l'appareil ou aussi à cause de la surface sur laquelle se trouve l'image, ce qui implique que les photos ont un éclairage pas souvent constant sur toute l'image, d'où la création de zones d'ombre et de zones de fort éclairage (Fig. 2).

Les déformations de perspective Les photos étant prises sur des journaux ou des magazines, les déformations de la grille carrée en trapèze sont quasiment inévitables dûes à l'angle de la prise de vue qui ne sera jamais exactement perpendiculaire au support [14]. Il en résulte que les caractères les plus éloignés semblent toujours plus petits et déformés.

La rotation Les images ne seront jamais parfaitement droites et seront le plus souvent soumises à une légère rotation de quelques degrés [13] [1]. Il existe aussi le problème des utilisateurs qui prennent des photos en pensant être bien droits mais la photo est prise à l'envers ou avec une rotation de 90 degrés.

Le zoom et le focus Les appareils-photos des téléphones portables sont conçus pour fonctionner à partir d'une certaine distance. Donc si des images sont prises de trop près, le focus sera mauvais et l'image sera de plus en plus floue [4]. Suivant la qualité des photos et la distance à laquelle les photos sont prises, la grille et les chiffres ne seront pas vraiment lisibles et nets, ce qui risque de modifier fortement la tâche lors de la reconnaissance des chiffres (Fig. 3).

Les surfaces non planes Les pages de livres ou de magazines ne sont jamais planes et sont le plus souvent courbées, ce qui entraînera des déformations lors de la prise de la photo. Il se peut aussi que la page où se trouve le sudoku soit abîmée, pliée ou mouillée ce qui accentuera les déformations sur la grille et les chiffres (Fig. 4 et Fig. 5).

Distortion due aux lentilles Plus une photo est prise trop près de la source, plus les risques d'une distortion sur les contours de l'image grandissent. Vu que la plupart des appareils photos des téléphones portables possèdent une lentille de basse qualité, la distortion qui en résulte peut être fortement amplifiée (Fig. 6).

La compression A cause des ressources limitées qu'il existe sur les téléphones portables, la plupart des images prises par un appareil-photo de téléphone portable est compressée. Il en résulte une perte d'informations car la compression est optimisée pour des images de paysages mais pas pour l'analyse de documents.

FIG. 2 – Illustration d'une mauvaise lumière sur une image

FIG. 3 – Image prise sans focus

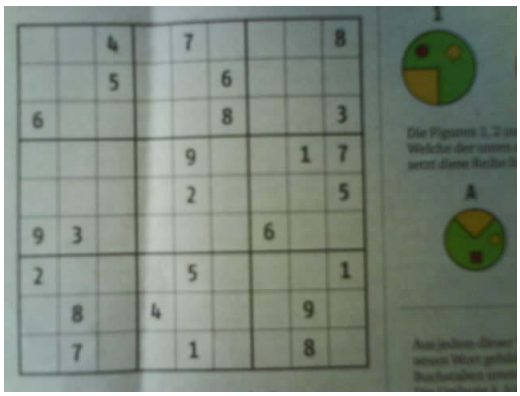

FIG. 4 – Sudoku sur une surface pliée

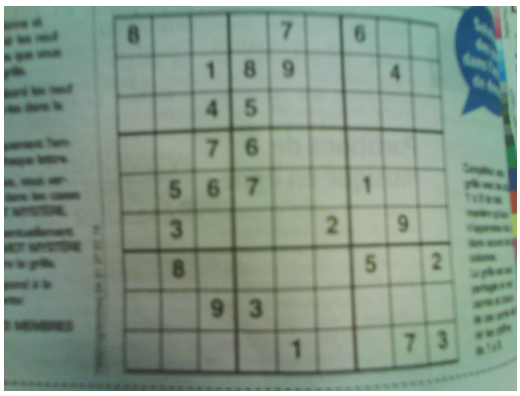

FIG. 5 – Sudoku sur une surface bombée

FIG. 6 – Distortion sur une image effet oeil de poisson

2.2 Redimensionnement des images

Les photos prises par un téléphone portable atteignent 5 MegaPixels ce qui représente une image de la taille de 2048 x 1536 pixels. Normalement lors de l'envoi d'une image avec la technologie MMS depuis son téléphone, ce dernier redimensionne l'image pour diminuer la taille du fichier à envoyer. Cependant cette étape n'est pas obligatoire et s'il le désire, l'utilisateur peut envoyer une image en grande résolution. D'où la nécessité d'effectuer un redimensionnement des images avant tout traitement, cela pour des raisons d'optimisation du temps de calcul sur les images et de normalisation de l'image.
Les images subissent un redimensionnement proportionnellement pour obtenir une largeur de 640 pixels. La plupart des images aura automatiquement une hauteur de 480 pixels car les dimensions des photos prises restent le plus souvent une homothétie d'une image 640 x 480 (Fig. 9).

2.3 Égalisation d'histogramme

L'égalisation d'histogramme (Fig. 7) est une transformation des niveaux de gris dont le principe est d'équilibrer le mieux possible la distribution des pixels dans la dynamique (Idéalement, il faut obtenir un histogramme plat).

2.4 Normalisation d'histogramme

La normalisation d'histogramme (Fig. 8) est une transformation affine qui a pour but d'harmoniser la répartition des niveaux de gris des pixels de l'image, de manière à tendre vers un même nombre de pixels pour chacun des niveaux de l'histogramme.

Cette opération vise à augmenter les nuances dans l'image.

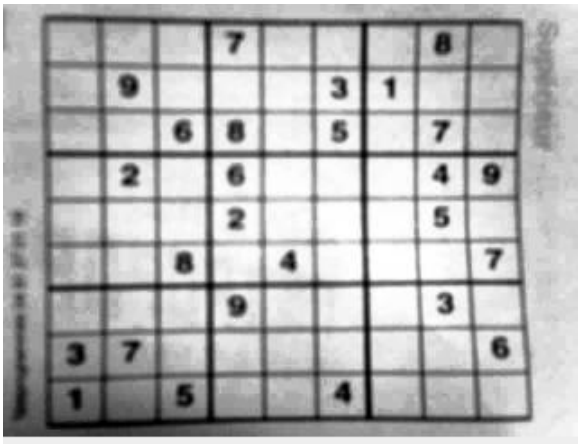

FIG. 7 – Résultat de l'égalisation d'histogramme

FIG. 8 – Résultat de la normalisation d'histogramme

2.5 Transformation en niveau de gris

Pour pouvoir travailler avec les images il est indispensable de faire une transformation en niveau de gris (Fig. 10).

Pour les images couleurs, un pixel dispose généralement des trois composantes RGB (en anglais : Red, Green, Blue ; en français : Rouge, Vert, Bleu). Un pixel gris a ses trois valeurs RGB identiques. Une méthode simple pour convertir une image couleur en niveau de gris pourrait être de calculer la moyenne des trois composantes RGB et d'utiliser cette valeur moyenne pour chacune des composantes.

$Gris = \frac{(Rouge+Vert+Bleu)}{3}$

La C.I.E (Commission Internationale de l'éclairage) propose de caractériser l'information de luminance (la valeur de gris) d'un pixel par deux formules :

- Dans sa recommandation 709, qui concerne les couleurs "vraies" ou naturelles :
  $Gris = 0.2125 \cdot Rouge + 0.7154 \cdot Vert + 0.0721 \cdot Bleu$

- Dans sa recommandation 601 pour les couleurs non-linéaires, c'est-à-dire avec correction du gamma (image vue à partir d'un écran vidéo)
  $Gris = 0.299 \cdot Rouge + 0.587 \cdot Vert + 0.114 \cdot Bleu$

Ces formules rendent compte de la manière dont l'oeil humain perçoit les trois composantes, rouge, vert et bleu de la lumière. Pour chacune d'elles, la somme des 3 coefficients vaut 1. On remarquera la forte inégalité entre ceux-ci : une lumière verte apparaît plus claire qu'une lumière rouge, et encore plus qu'une lumière bleue.

2.6 Binarisation des images

Pour la binarisation des images, quatre différentes méthodes ont été testées afin d'obtenir le meilleur résultat :

- La méthode de binarisation d'Otsu.
- La méthode de binarisation de Niblack.
- Une méthode de binarisation basée sur Niblack.
- La méthode de binarisation de Sauvola.

2.6.1 La méthode de binarisation d'Otsu

Le but de cet algorithme est la binarisation d'images en niveaux de gris (Fig. **??**). Ceci revient à séparer les pixels de l'image en deux classes, la première ayant un niveau maximal (typiquement 255) et la seconde un niveau minimal (0). Cette méthode de binarisation nécessite au préalable le calcul de l'histogramme. Puis, la séparation en deux classes est effectuée. Il est important de noter d'ores et déjà qu'il est assez simple d'étendre ce qui va suivre à un nombre de classes plus important.

Le calcul de l'histogramme est très simple.

Un tableau *histo* est initialisé avec des 0. Généralement, ce tableau est constitué de 255 cases correspondant aux 255 niveaux de gris d'une image.
Ensuite, si p(i,j) représente la valeur du pixel au point (i,j), on balaye toute l'image et on compte le nombre de fois où un niveau de gris apparaît.

D'après l'article [10] ; la séparation se fait à partir des moments des deux premiers ordres : la moyenne et l'écart-type. Pour que le procédé soit indépendant du nombre de points dans l'image N, l'histogramme : $p_i = n_i/\mathrm{N}$ est normalisé où n_i représente le nombre de pixels de niveau i. Les deux moments utilisés peuvent, alors être calculés : mu(k) = sum(i=1,k) i*p_i w(k) = sum(i=1,k) p_i. L'expression sum(i=1,k) représente la somme de i=1 à k.

Nous notons mu_T=mu(256), où 256 est le nombre total de niveaux de gris.

Si w_0 est appelé la probabilité de la classe C_0 et w_1 la probabilité de la classe C_1, alors :

w_0 = w(k*) où k* représente le niveau de seuil et w_0 = 1 - w(k*).

Si nous notons de même mu_1 et mu_0 avec : mu_0 = mu(k*)/w(k*) mu_1 = (mu_T-mu(k*))/(1-w(k)).

Or l'image totale conserve certaines propriétés, d'où nous pouvons tirer les relations :

$$w_0 mu_0 + w_1 mu_1 = mu_T w_0 + w_1 = 1.$$

En introduisant un paramètre pour évaluer la qualité du niveau de seuillage, nous obtenons : $s^2 = w_0 w_1 (mu_1 - mu_0)^2$.

La valeur précédente est fonction de k. Nous calculons donc cette valeur pour les 256 niveaux de gris de l'image. (en fait les valeurs 0 et 255 qui correspondent à affecter tous les pixels à la même classe peuvent être déjà enlevées).

A partir de w(k) et mu(k), nous calculons donc : $s^2(k) = w(k)(1-w(k))(mu_T w(k) - mu(k))^2$.

La valeur du seuil k* est obtenue pour le maximum de s^2.

Il ne reste plus qu'à comparer la valeur de tous les pixels de l'image au seuil ainsi trouvé.

2.6.2 La méthode de binarisation de Niblack

D'après l'article [7], l'idée de la méthode de binarisation de Niblack est de faire varier le seuil dans l'image en fonction des valeurs de la moyenne locale et de l'écart type local (Fig. 12). Le seuil calculé pour le pixel (x,y) est :

$T(x, y) = m(x, y) + k.s(x, y)$

où $m(x, y)$ et $s(x, y)$ sont respectivement la moyenne et l'écart type calculés dans un voisinage local de (x, y).

La taille du voisinage doit être suffisamment petite pour préserver les détails locaux, mais suffisamment large pour supprimer le bruit. La valeur de k est utilisée pour ajuster ce paramètre.

Si nous notons $f(x, y)$ le niveau de gris dans un point (x, y), alors l'écart type local $s(x, y)$ dans un voisinage de taille $(2k_1 + 1) \cdot (2k_2 + 1)$ autour de (x,y) peut être calculé comme :

$s^2(x, y) = \frac{1}{(2k_1+1)(2k_2+1)} \sum_{m=-k_1}^{k_1} \sum_{n=-k_2}^{k_2} (f(x + m, y + n))^2 - (m(x, y))^2$

où $m(x, y)$ est la valeur moyenne de $f(x, y)$ dans le voisinage

$m(x, y) = \frac{1}{(2k_1+1)(2k_2+1)} \sum_{m=-k_1}^{k_1} \sum_{n=-k_2}^{k_2} f(x + m, y + n)$

2.6.3 Méthode de binarisation basée sur Niblack

Cette méthode reprend la binarisation de Niblack mais effectue un test sur l'écart-type des pixels dans la région locale de binarisation. Si cette région est en dessus d'un seuil, tous les pixels en blanc sont validés et nous passons à une nouvelle zone, cette méthode permet ainsi de nettoyer encore plus le bruit sur une image (Fig. 13.

2.6.4 La méthode de binarisation de Sauvola

Sauvola [12] propose un seuil :

$T(x,y) = m(x,y) \cdot [\frac{1+k \cdot s(x,y)}{R-1}]$

où R est le domaine dynamique de la variance et où k prend des valeurs positives. La multiplication des deux termes par la moyenne locale a pour effet d'amplifier la contribution de la variance de manière adaptative. Si nous considérons par exemple un texte foncé sur un fond clair, mais avec du bruit, m(x,y) fait décroître la valeur du seuil dans les régions du fond (Fig. **??**sauvola). L'effet de cette méthode est d'effacer d'une manière efficace le bruit dans une image seuillée.

2.6.5 Conclusions et résultats sur la binarisation des images

Comparons à présent les 4 méthodes de binarisation citées ci-dessus pour déterminer avec quelle méthode la meilleure solution est obtenue.

Il est important de souligner que pour la suite du projet nous devons trouver une image binarisée la plus propre possible au niveau du bruit et la moins sensible au variation de lumière pour une meilleure détection de la grille.
Au début du processus nous nous retrouvons avec une image en couleur qui est préalablement redimmensionnée (Fig. 9), cette image est transformée en niveau de gris et subit une normalisation d'histogramme (Fig. 10).

Si une binarisation avec la méthode d'Otsu est appliquée (Fig. 11), cette méthode est très sensible aux changements d'intensité dans le fond de l'image et il manque de nombreuses informations par rapport à l'image de départ ; il manque des morceaux de grille ce qui ne sera pas très optimal ni pour la détection de la grille ni pour la reconnaissance des chiffres. La présence de texte même très léger sur les bords de la grille se retrouve transformée en grosse tâche noire ce qui pourrait facilement biaiser la détection des verticales.
En conclusion cette méthode nettoye bien le fond de l'image mais est beaucoup trop sensible au variation d'intensité des pixels. Donc ce n'est pas une bonne solution pour notre problème.

Testons à présent une binarisation avec la méthode de Niblack (Fig. 12).
Cette méthode est légèrement meilleure que celle d'Otsu reste aussi sensible à la variation de la luminosité dans le fond. Les textes sur les bords de la grille sont moins marqués qu'avec Ostu . Cependant cette méthode n'est pas complètement satisfaisante vu le que le résultat n'est de loin pas optimal.

De la méthode de binarisation de Niblack, nous avons essayé une nouvelle technique de binarisation basée sur Niblack, seulement en jouant avec les écarts-types dans la fenêtre de travail(Fig. 13).
Cette solution est beaucoup plus concluante que les deux précédentes ; toute la grille est visible et cette binarisation est très peu sensible aux variations des pixels dans le fond de l'image. Cependant un excès de bruit apparaît dans le fond de l'image. Ce bruit n'est pas très important et pourrait ne pas affecter la détection de la grille mais il serait préférable d'avoir le moins de bruit possible

pour la reconnaissance des chiffres.

La dernière méthode testée est la binarisation avec la méthode de Sauvola (Fig. 14). Cette méthode n'est ni sensible au bruit dans le fond, ni sensible aux variations de lumière et donne une image binarisée propre avec la grille et ses chiffres dans son intégralité. Le bruit dans le fond est très faible voir inexistant et même les textes sur les bords restent de taille acceptable, presque comme dans l'image initiale.

L'image binarisée ainsi obtenue est tout à fait acceptable pour une détection de la grille du sudoku.

Cette méthode est la meilleure méthode trouvée pour binariser une image de grille de sudoku. Donc le meilleur traitement a effectuer pour travailler avec une image de grille de sudoku sera :

1. Redimentionner l'image sans la déformer, pour diminuer le temps de calcul des traitements de l'image.
2. Transformer en niveau de gris afin de pouvoir travailler sur une image ayant moins de paramètres mais sans perdre d'informations sur la composition de l'image.
3. Normaliser d'histogramme dans le but d'harmoniser la répartition des niveaux de gris.
4. Binariser avec la méthode de Sauvola.

Fig. 9 – Image en couleur redimensionnée

FIG. 10 – Résultat niveau de gris + normalisation d'histogramme

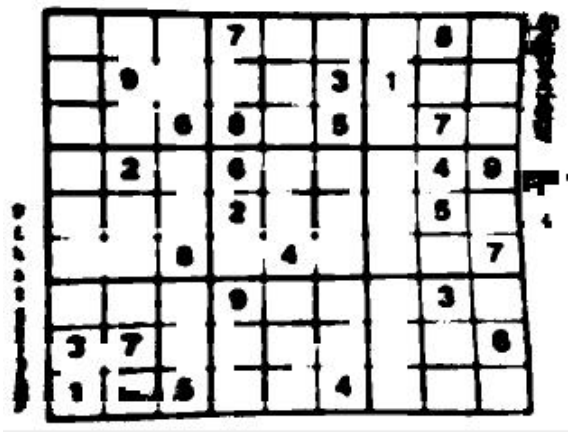

FIG. 11 – Binarisation avec la méthode d'Otsu

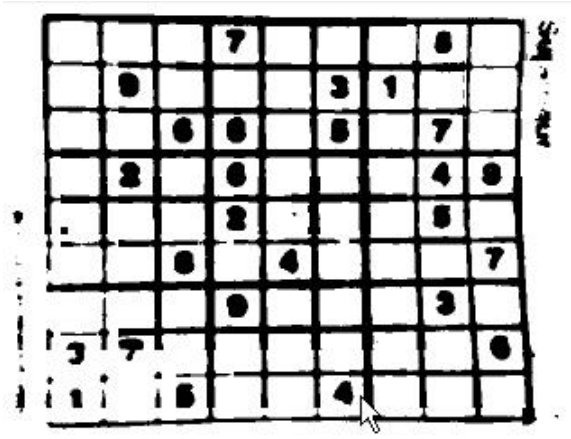

Fig. 12 – Binarisation avec la méthode de Niblack

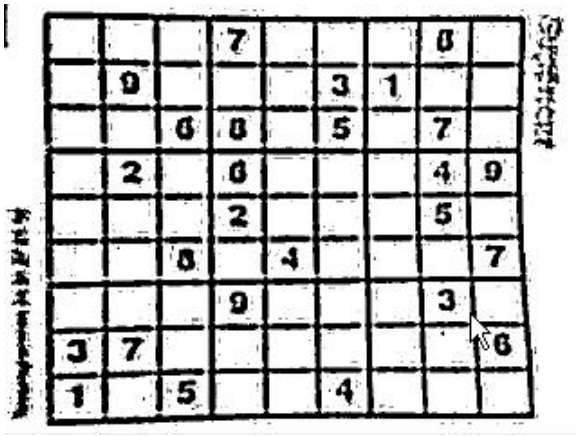

Fig. 13 – Binarisation avec la méthode de Niblack améliorée

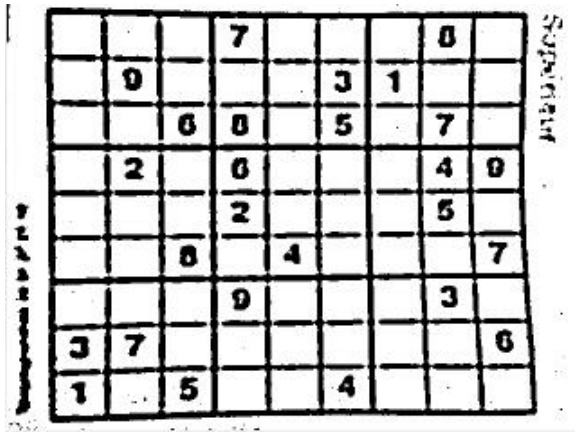

Fig. 14 – Binarisation avec la méthode de Sauvola

2.7 La dilatation

Une fois la binarisation effectuée, il faut détecter la grille et les cellules des sudokus Cependant la binarisation laisse apparaître des petites zones de pixels noirs manquantes dans les segments de la grille des sudokus. Ces trous peuvent empêcher la détection des cases. Il est donc impératif d'avoir des cases bien fermées et de combler ces trous. Pour ce faire, une dilatation de pixels est utilisée à la fois verticale et horizontale ce qui va permettre d'être sûr que les cases seront bien fermées. La dilatation s'effectue sur 5 pixels, c'est à dire que si un pixel noir est détecté , il est dilaté de cinq pixels verticalement et horizontalement.
Le résultat de la dilatation donne une image plus marquée et plus épaisse mais cela est un avantage pour la détection des grilles (Fig. 15).

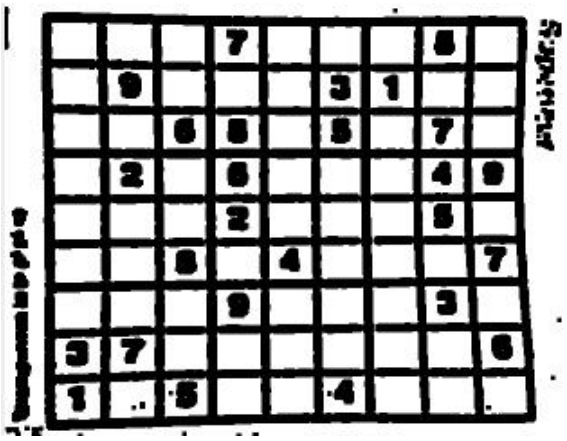

Fig. 15 – Dilatation d'une image binarisée

3 Algorithmes de détections

La détection de la grille est effectuée sur une image binarisée. Une image en niveau de gris présente des pixels ayant des valeurs de 0 à 255, tandis qu'une image binarisée ne possède que deux niveaux ; noir et blanc soit 0 ou 255.

Nous allons, à présent, exposer et comparer trois différentes méthodes dans le but de détecter la grille dans une image contenant un sudoku et un algorithme pour la détection des cases.

3.1 Détection de la grille

3.1.1 Détection avec un modèle de grille

Dans un premier temps nous avons essayé de créer un modèle de grille. Le score de chaque essai d'ajustage de la grille est obtenu en calculant le nombre de pixels noirs que notre modèle a en commun avec l'image binarisée de la grille du sudoku. Un pourcentage de conformité est obtenu et théoriquement, le modèle ayant le meilleur score sera la bonne modélisation de la grille.

A cause de la faible résolution et de la mauvaise qualité des images, les segments de la grille du sudoku sont flous. Ensuite ils ne sont plus formés de droites mais de courbes à cause des déformations dues aux appareils-photos et aux optiques des téléphones portables et surtout ils ne se trouvent plus à égale distance à cause de ces déformations.
Cette méthode est extrêmement coûteuse en temps de calcul car le modèle de grille doit avoir les fonctions basiques suivantes :

- Translation de la grille sur l'image.
- Dilatation de la grille en longueur et en largeur.
- Rotations de la grille.
- Déplacement individuel de chaque segment.
- Déplacement individuel de chaque noeud de la grille.

La recherche du meilleur modèle sur un optimum de score donné par le pourcentage de pixels noirs requiert un temps de calcul compris entre 30 et 40 secondes et le plus souvent donne une mauvaise solution. Le résultat n'est pas bon car les cases peuvent être remplies de nombres qui sont eux aussi composés de pixels. Les grilles trouvées sont très mal ajustées voire pas du tout en concordance avec la grille de l'image, bien que le score de notre modèle soit le plus haut. Ce modèle cherche à se calquer sur les pixels noirs et comme la grille n'est pas droite cela reste très difficile. Il faudrait implémenter une déformation des segments mais cela ajouterait encore du temps de calcul.
L'exemple de la figure 16 démontre que la grille n'est pas bien détectée et le texte qui se trouve hors et dans les cases est considéré comme étant un segment. Cela est particulièrement dû au fait que les segments ne sont pas des droites. Donc même si notre modèle est parfaitement ajusté sur une verticale ou sur une horizontale, il n'obtiendra même pas un score de 100%. Cela rend la détection très difficile.
Ce système fonctionne pour 60% des images ce qui est très faible surtout que

pour arriver à ces résultats, il faut un temps de calcul beaucoup trop long.

Pour fonctionner correctement, cette méthode devrait être utilisée sur une image non déformée d'une grille de sudoku. Il faudrait avoir une image scannée ou procéder à un traitement de l'image qui corrigerait les déformations dues aux lentilles. De tels traitements d'image existent mais il serait préférable de trouver une autre solution avant d'ajouter encore un traitement de l'image qui annulerait les déformations.

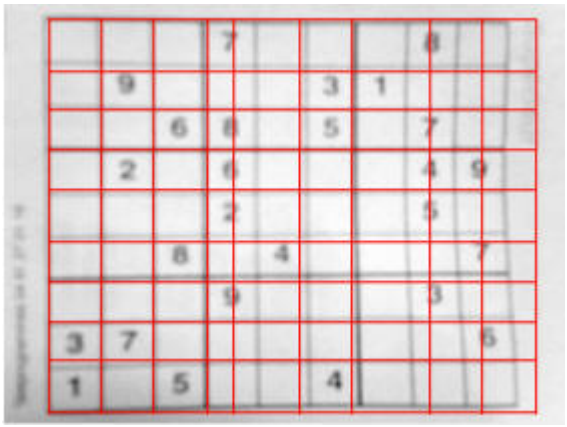

FIG. 16 – Détection par modèle de grille

3.1.2 Détection avec éliminations des mauvaises valeurs

Une approche différente pour la détection de la grille consiste à effectuer la projection de la valeur des pixels de l'image sur les axes X (Fig. 17)et sur l'axe de Y (Fig. 18) ; en supposant que l'axe des X représente la largeur de l'image et l'axe des Y la hauteur de l'image. Le résultat de ces projections comporte des pics aux emplacements où il y a le plus de pixels noirs sur l'image donc théoriquement à l'emplacement des verticales et des horizontales de la grille.
Cependant il y beaucoup de bruit autour de l'image et certaines rangées de chiffres inclus dans la grille ont des pics plus élevés que les verticales, cela est en partie dû aux déformations que la grille subit avec les appareils-photos des téléphones portables. Il faut aussi préciser que si une image a subi une légère rotation les pics seront plus aplatis et il est évident que des pics plus forts apparaîtront sur des rangées de chiffres ou sur du texte autour de la grille de sudoku. D'où l'idée de repérer les 12 pics les plus élevés du graphique et non pas les 10 pics les plus élevés. La grille du sudoku est dans tous les cas contenue dans cette nouvelle grille composée de 12 verticales et 12 horizontales.
Vu que la grille du sudoku est contenue dans cette nouvelle grille formée de 12 verticales et horizontales (Fig. 19), il faut trouver un moyen de retirer les 2 mauvaises verticales et horizontales de cette nouvelle grille. Ces verticales et horizontales de trop peuvent être repérées plus ou moins facilement car certaines constantes qui définissent des grilles sont connues :

- Une grille de sudoku comporte dix verticales et horizontales
- La distance entre les droites est constante
- La fréquence d'apparition des droites est constante.
- L'emplacement des droites ne peut dépasser une certaine limite (on doit avoir 10 droites et on connaît la taille de l'image donc on peut en déduire si des droites sorte de l'image)

Deux observations sur le comportement des mauvaises droites de la grille ont été faites :

- Dans la grille une mauvaise détection s'effectue le plus souvent entre deux segments sur des chiffres
- Hors de la grille, la mauvaise détection est due au texte autour de la grille

De ces observations un algorithme d'élimination est implémenté :

- Prendre la moyenne de la valeur des espaces entre les segments de la grille.
- Garder les segments qui sont dans un rayon proche de cette moyenne.
- Pour les mauvais segments, la distance entre trois segments consécutifs doit être calculée..
- Les bons segments sont gardés et si deux mauvais segments n'ont pas été éliminés, cet algorithme sera recommencé avec un rayon plus petit.

Cette méthode est assez rapide et performante. Cependant seulement 70% des images retournent de bons résultats.
L'explication de ce faible taux de détection est dû au fait que les images ne comportant pas de bruit autour de la grille et comportant de très petites déformations,

ne répondent plus aux règles établies précédemment. L'algorithme de détection fonctionne seulement pour des images mauvaises avec du bruit et du texte autour de la grille. Car il est élaboré en observant la réaction de la grille comportant 12 verticales et horizontales sur de mauvaises images. Sur des images de bonne qualité la prédictivité du système est quasiment impossible.

En conclusion cet algorithme serait efficace si toutes les images traitées étaient mauvaises, or ce n'est heureusement pas le cas : la base de donnée, dont nous parlerons plus tard dans ce rapport, est aussi composée d'images de bonne qualité. Nous ne pouvons pas imaginer que tous les utilisateurs envoient des images déformées et de mauvaise qualité. Un bon système de détection devrait fonctionner dans les cas de figures bonnes et mauvaises, pas seulement dans un seul des deux cas. Cette méthode n'est pas utilisable pour détecter la grille vu les mauvais résultats obtenus sur des images de bonne qualité.

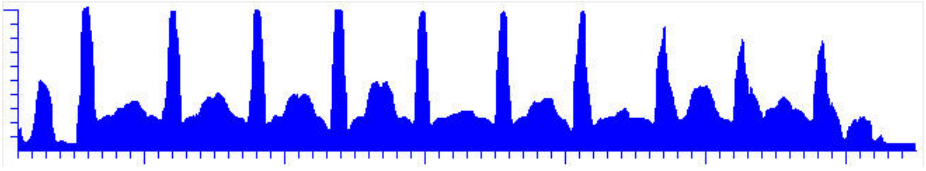

FIG. 17 – Projection des pixels noirs sur l'axe des X d'une image contenant une grille de sudoku

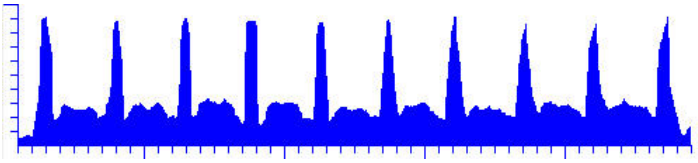

FIG. 18 – Projection des pixels noirs sur l'axe des Y d'une image contenant une grille de sudoku

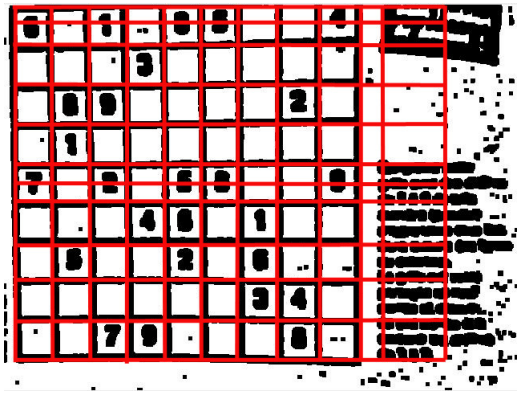

FIG. 19 – Grille de sudoku incluse dans la nouvelle grille formée de 12 verticales et horizontales

3.1.3 Détection avec les composantes connexes

La *composante connexe* en traitement d'image représente une agrégation de pixels connectés sur une image.
La technique de marquage de composantes connexes est basée sur une détection des contours des composantes. Suite à cette détection des contours, les pixels d'une composante donnée sont marqués par un même label suivant leur relation de connexité (4 ou 8 connexes) . L'exemple ci-dessous donne un exemple de résultat d'un marquage de composantes en 4-connexité.

Lorsque le système détecte une composante connexe, il renvoie la zone rectangulaire la plus petite qui peut contenir cette composante connexe trouvée (Fig. 20).

0	0	0	0	0	0	0	0	0
0	1	1	0	0	3	3	3	0
0	0	1	0	0	0	3	3	0
0	1	0	0	0	3	0	3	0
0	0	0	0	0	0	0	0	0
0	0	0	2	2	0	0	0	0
0	0	0	0	2	0	0	0	0
0	0	0	2	0	0	0	0	0
0	0	0	0	2	0	0	0	0

FIG. 20 – Exemple de marquage de composantes en 8-connexité avec la zone retournée en rouge

La dernière méthode testée est de rechercher la zone rectangulaire contenant la plus grande composante connexe noire, donc celle ayant la plus grande aire

dans l'image. Si nous observons les images de grilles de sudoku, cette dernière est constituée d'une grille dont le contour est défini par des pixels noirs qui sont tous connectés entre eux même si l'image est déformée par les différentes lentilles des appareils-photos. Cette méthode permet alors d'isoler la grille du reste de l'image (Fig. 21) avec un taux de réussite de plus de 99%. Le seul type d'image qui reste mal détecté est une image découpée et posée sur un fond de couleur foncé donc ayant un cadre artificiel (Fig. 22). Ce cas n'est pas vraiment relevant car le but du projet est la détection de grilles prises en photo dans des journaux ou des magazines.

Une fois la grille du sudoku isolée du reste de l'image, nous pouvons utiliser la détection de la grille du sudoku avec la méthode des 10 plus grands pics sur les projections de la valeur des pixels noirs de l'image sur les axes X et Y, car cette fois il n'y a absolument plus aucun bruit autour de la grille (Fig. 23).

Cette méthode est très efficace et beaucoup moins gourmande en ressources que les deux précédentes méthodes. Il est même possible de détecter la zone connexe contenant la grille du sudoku à partir d'une image en niveau de gris d'où un gain de temps considérable.

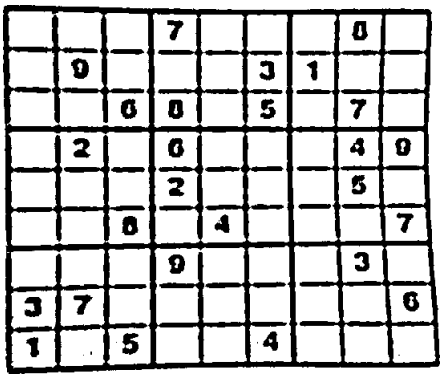

FIG. 21 – Grille isolée après détection des pixels connectés

3.2 Détection des cases

La détection des cases semble être inutile vu que la grille est détectée mais il faut remarquer que la détection de la grille se base sur les pics des projections de la valeur des pixels noirs sur les axes X et Y. Et que les images de grilles de sudoku étaient déformées à cause de la qualité et des objectifs des appareils-photos des téléphones portables. Les chiffres se trouvent bien dans leurs cases respectives mais le plus souvent ces chiffres contenus à l'intérieur sont mal centrés. Il se peut même de trouver un partie de segment de grilles ou un cadre (Fig. 24 et Fig. 25).
Il est nécessaire pour la suite du projet d'isoler le mieux possible le chiffre du reste de la case. Pour cela, il est impératif de détecter si la case est vide et d'utiliser un algorithme d'isolation du chiffre.

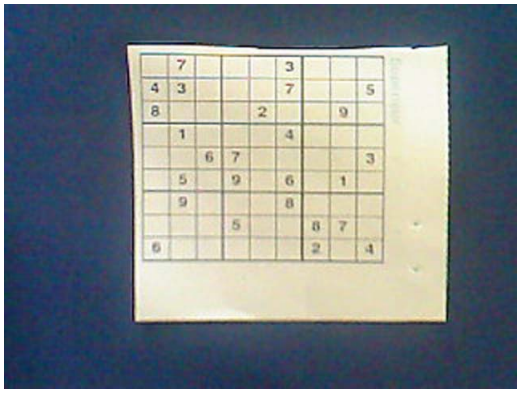

FIG. 22 – Image découpée

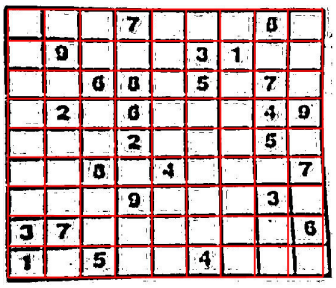

FIG. 23 – Détection de grille isolée

FIG. 24 – Case mal détectée

FIG. 25 – Case mal détectée

3.2.1 Vérification des cases vides

Les grilles du sudoku sont constituées de cases contenant un chiffre ou de cases vides. Il est important de détecter si la case est vide ou non pour la reconnaissance du chiffre par le réseau de neurones. Pour ce faire nous calculons dans chaque case en niveau de gris l'écart-type de la valeur de tous les pixels contenus dans la case. Si cette valeur est en dessous d'un certain seuil, la case est considérée comme vide et une case blanche est retournée au système.

3.2.2 Isolation du chiffre

Les cases contenant des chiffres doivent être nettoyées. C'est à dire, il faut isoler le chiffre du reste de la case en utilisant l'algorithme suivant :

- Détection de la plus grande zone connexe blanche dans la case, pour enlever le cadre autour du chiffre.
- Détection de la plus grande zone noire dans la région trouvée précédemment, cela nous permet d'isoler le chiffre.
- Création d'une case blanche carrée de côté égal au maximum entre la longueur et la largeur de la dimension du chiffre.
- Inclure et centrer le chiffre dans la case blanche.
- Redimensionnement de la case en 20 X 20 pixels.

En effectuant cet algorithme, nous obtenons une série de cases homogènes avec des chiffres non déformés et centrés. De cette manière le type de caractère utilisé pour le chiffre devient obsolète car tous les chiffres ont la même dimension et ne sont pas déformés.(Fig. 26).

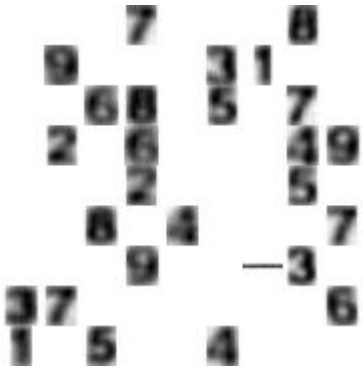

FIG. 26 – Résultat de la détection des cases vides et de l'isolation des chiffres

4 Reconnaissance des chiffres

La reconnaissance des chiffres va se faire avec l'utilisation d'un réseau de neurones. Nous allons d'abord faire un bref résumé sur les réseaux de neurones et définir la création du réseau de neurones pour ce projet.

4.1 Bref rappel sur les réseaux de neurones

Comme leur nom l'indique, les réseaux de neurones sont ainsi organisés autour d'un ensemble de neurones, connectés entre eux par des liaisons affectées de poids. Nous disposons initialement d'une base de données constituée d'un ensemble de données entrées et sorties et nous souhaitons utiliser cette base de données pour entraîner une algorithme à reproduire les associations constatées entre les entrées et les sorties de l'échantillon.

Les neurones sont connectés entre eux par des liaisons affectées de poids. Ces liaisons permettent à chaque cellule de disposer d'un canal pour envoyer et recevoir des signaux en provenance d'autres neurones du réseau. Chacune de ces connexions reçoit un poids, qui détermine son impact sur les neurones qu'elle connecte. Chaque neurone dispose ainsi d'une entrée, qui lui permet de recevoir de l'information d'autres neurones, et aussi d'une fonction d'activation, qui est dans les cas les plus simple, une simple identité du résultat obtenu par l'entrée et la sortie. Ainsi, pour un réseau de neurones avec N cellules dans la première couche, notées C(1),...,C(N) , et N poids affectées aux liaisons et notés w(1),...,w(N) l'entrée d'un neurone de la seconde couche sera généralement une somme pondérée des valeurs de sortie des neurones précédents :

$$X = w(1) * C(1) + w(2) * C(2) + w(3) * C(3) + ... + w(N) * C(N)$$

Pour obtenir la valeur de sortie Y du neurone concerné, nous pouvons utiliser une fonction d'activation identité du type :

$$Y = d * X$$

Le *perceptron multicouches* est sans doute le plus simple et le plus connu des réseaux de neurones. La structure est relativement simple : une couche d'entrée, une couche de sortie et une ou plusieurs couches cachées. Chaque neurone n'est relié qu'aux neurones des couches précédentes, mais à tous les neurones de la couche précédente. La fonction d'activation utilisée est en général une somme pondérée (Fig. 27).

4.2 Architecture générale du réseau de neurones

Le réseau de neurones utilisé dans ce projet est un *perceptron multi-couches*, cette architecture est particulièrement bien adaptée à la classification d'ensemble de vecteurs d'entrée en ensemble de vecteurs de sortie correspondants. Le *perceptron* est normalement défini comme ayant une couche d'entrée, une couche cachée et une couche de sortie.

L'implementation du réseau de neurones suit la description du livre de Duda/Hart/Stork [11] en utilisant une fonction d'apprentissage de *rétropropagation*

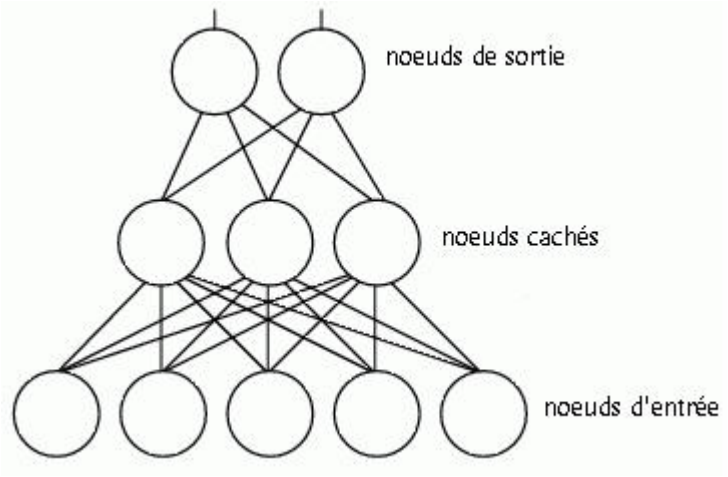

FIG. 27 – Schéma type d'un perceptron à trois couches

avec la méthode des moments. La fonction d'activation par défaut est la sigmoïd centrée ; tangente hyperbolique.

4.2.1 La couche d'entrée

La première couche, appelée couche d'entrée, reçoit les neurones contenants les caractéristiques extraites des chiffres. Ces entrées seront transmises au réseau par l'intermédiaire des fichiers d'entraînement et de validation. Cette couche comporte 88 neurones qui seront détaillés plus tard.

4.2.2 La couche cachée

La seconde couche est se nomme la couche cachée, en ce sens qu'elle n'a qu'une utilité intrinsèque pour le réseau de neurones et n'a pas de contact direct avec l'extérieur. La fonction d'activation utilisée est la tangente hyperbolique. Le choix de sa taille n'est pas implicite et doit être ajusté. En général, il est recommandé de commencer par une taille moyenne du nombre de neurones dans les couches d'entrée et de sortie. Pour obtenir de bons résultats, il faut tester le plus de tailles possibles.

4.2.3 La couche de sortie

La troisième couche est appelée couche de sortie. Elle donne le résultat obtenu après compilation par le réseau de neurones des données entrées dans la première couche. Le réseau de neurone devra être seulement en mesure de déterminer la valeur d'un chiffre dans une case donc il y a 10 neurones pour la couche de sortie chaque neurone représentant une valeur allant de 0 à 9.

4.3 Construction des échantillons pour les ensembles d'entraînement et de validation

Chaque image comporte 81 cases. Notre base de données est composée de 200 images. Trois quarts de notre base de données est utilisés pour entraîner le système et le reste pour la validation. Dans notre cas la dimension de l'ensemble d'entraînement est de 12'069 échantillons et celle de l'ensemble de validation de 4'132 échantillons.

4.3.1 Création d'une base de données d'images

Pour tester et entraîner le réseau de neurones il est impératif d'avoir une base de données pour obtenir des échantillons. Les images de sudoku ont été prises avec des téléphones portables de différentes marques et de différents modèles :

- Sony-Ericson S500i
- Sony-Ericson W880i
- Sony-Ericson K810i
- Sony-Ericson K800i
- Sony-Ericson W810i
- Nokia E65
- Nokia 6290
- Samsung SGH-D900i
- Motorola MOTORAZR V3
- Motorola MOTORAZR V3xx

4.3.2 Descriptif de la base de données d'images

Chaque image est répertoriée dans un fichier XML qui contient des informations sur l'image et sur la grille de sudoku (Fig. 28)

L'image est décrite selon les éléments suivants :

uri L'emplacement de l'image

camera Des informations sur la marque et le modèle du téléphones avec lequel l'image a été prise.

quality Une description quantitative du niveau de déformation, du niveau d'éclairage, de la rotation, du niveau de flou et des ombres sur l'image.

imageSpecification Des informations sur la taille de l'image, l'encodage et le nombre de bit par pixels qui composent l'image.

Le grille du sudoku est définie par les éléments suivants :

labelling Indication définissant si les entrées ont été effectuées par une machine ou manuellement.

gridPosition Indication sur la position de la grille du sudoku dans l'image, représentée par les coordonnées X et Y des pixels de l'image.

row Les valeurs des cases de la grille de sudoku, classées par lignes et colonnes

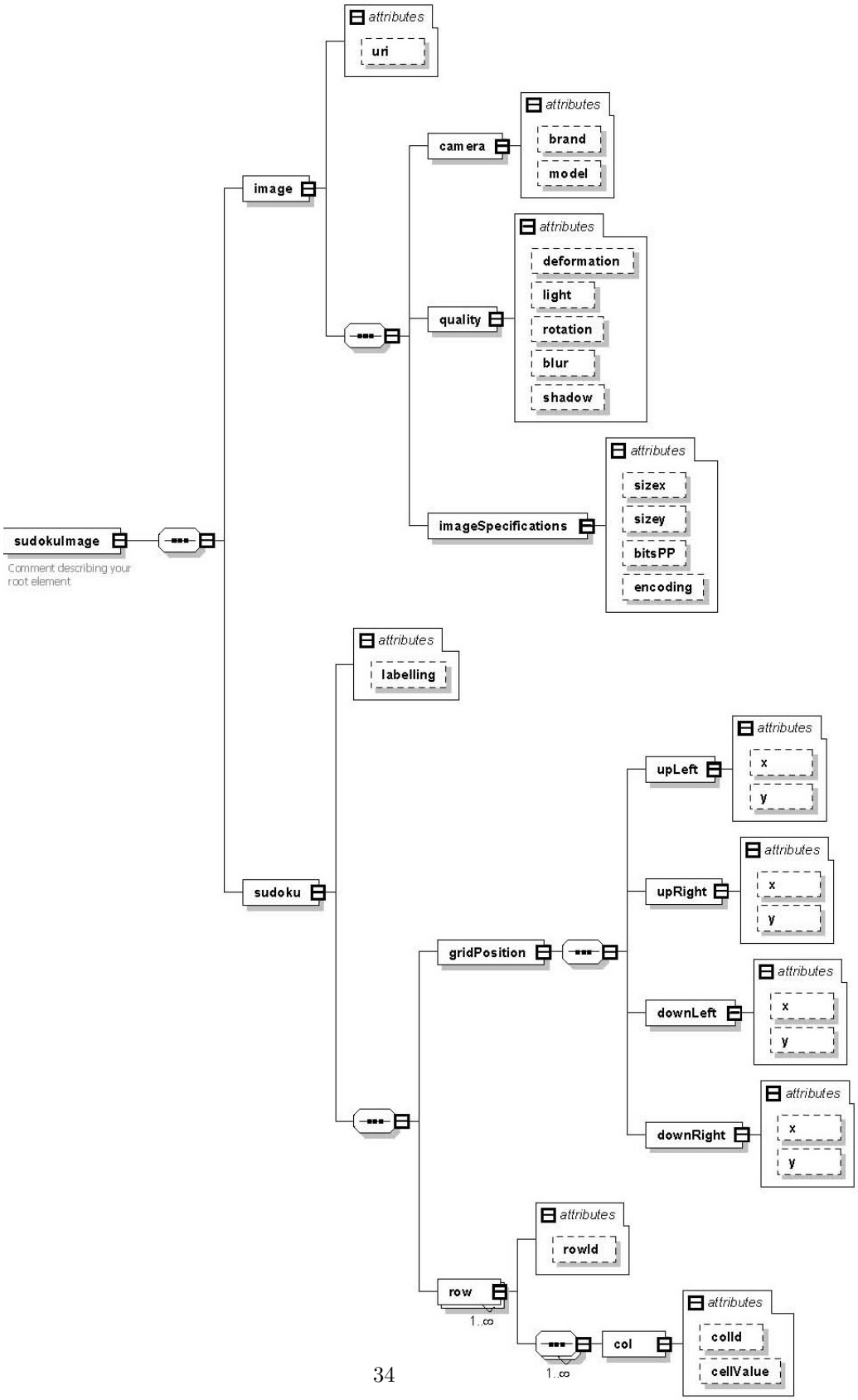

FIG. 28 – XML schéma d'une image de sudoku

4.3.3 Caractéristiques extraites

L'extraction de caractéristiques sur les cases comportant un chiffre (Fig. 29) est une étape importante car les chiffres sont le plus souvent flous à cause de la faible qualité des images. Il existe de nombreuses caractéristiques qui peuvent être extraites d'un échantillon pour effectuer la reconnaissance avec un réseau de neurones. Dans le cas des chiffres dans les cases de sudoku seulement les projections des pixels des cases, la dérivée de ces projections, la moyenne et l'écart-type sont utiles. Il n'est pas nécessaire d'extraire d'autres caractéristiques car le travail de centrage et de redimensionnement effectué à l'étape précédente normalise parfaitement les chiffres. Le réseau de neurones n'a alors pas besoin d'apprendre la translation, ni d'apprendre plusieurs styles de caractères différents pour chaque chiffre.

88 neurones sont utilisés en entrée pour permettre de recevoir les données concernant les caractéristiques extraites des échantillons :

- 20 neurones pour la projection verticale de l'image du caractère redimmensionnée sur une grille de 20 X 20 pixels
- 20 neurones pour la projection horizontale de l'image du caractère redimmensionnée sur une grille de 20 X 20 pixels
- 1 neurone pour la moyenne de la projection verticale
- 1 neurone pour la moyenne de la projection horizontale
- 1 neurone pour l'écart-type de la projection verticale
- 1 neurone pour l'écart-type de la projection horizontale
- 20 neurones pour la dérivée de la projection verticale de l'image du caractère redimmensionnée sur une grille de 20 X 20 pixels
- 20 neurones pour la dérivée de la projection horizontale de l'image du caractère redimmensionnée sur une grille de 20 X 20 pixels
- 1 neurone pour la moyenne de la dérivée de la projection verticale
- 1 neurone pour la moyenne de la dérivée de la projection horizontale
- 1 neurone pour l'écart-type de la dérivée de la projection verticale
- 1 neurone pour l'écart-type de la dérivée de la projection horizontale

FIG. 29 – Exemple d'une image d'où sont extraites les caractéristiques du chiffre 1

4.4 Entraînement du réseau de neurones

L'entraînement consiste tout d'abord à calculer les poids optimaux des différentes liaisons, en utilisant un échantillon. La méthode utilisée est la *rétropropagation* : les échantillons sont passés dans les cellules d'entrée et en fonction de l'erreur obtenue en sortie, les poids sont corrigés accordés aux pondérations. Il faut faire attention ne ne pas sur-entraîner le réseau de neurones pour éviter d'avoir de l'*overfitting*.

4.5 Résultats obtenus pour le réseau de neurones

Normalement l'optimisation d'un réseau de neurones s'effectue avec trois ensembles distincts ; un ensemble d'entraînement, un ensemble de validation et un ensemble pour les mesures de performances effectives. Dans notre cas nous n'avons que deux ensembles qui sont l'ensemble d'entraînement et l'ensemble de validation et les optimisations seront effectuées seulement sur l'ensemble de validation.

Le réseau de neurones étant défini par la description du livre de Duda/Hart/Stork [11] dispose de trois paramètres principaux qui sont :

- Le nombre de neurones sur la couche cachée
- Le taux d'apprentissage η.
- La valeur du moment de *rétropropagation* μ

Des tests sont effectués en faisant varier ces trois paramètres afin d'obtenir l'erreur la plus basse et le taux de reconnaissance le plus haut pour la reconnaissance.

4.5.1 Optimisation du nombre de neurones dans la couche cachée

Une série de tests a été effectuée pour définir le nombre de neurones dans la couche cachée. Ces tests permettent d'obtenir un pourcentage de reconnaissance de l'ensemble d'entraînement et de l'ensemble de validation. Comme on peut le voir sur la figure 30, un total de 70 neurones dans la couche cachée semble être la solution qui fournit le meilleur résultat ; le pourcentage de chiffres correctement détectés est de 99,28% pour l'ensemble d'entraînement et de 98,98% pour l'ensemble de validation.

La figure 31 représente l'évolution du taux de reconnaissance des chiffres sur 100 itérations du réseau de neurones. L'évolution sur l'ensemble d'entraînement est défini par la courbe rouge et l'évolution sur l'ensemble de validation par la courbe verte.

Hidden	Epoch	nu	mu	train error	train rate	valid error	valid rate
10	100	0.01	0.8	0.0032	0.9892	0.0065	0.9891
20	100	0.01	0.8	0.0027	0.9917	0.0049	0.9906
30	100	0.01	0.8	0.0028	0.9911	0.0053	0.991
40	100	0.01	0.8	0.0028	0.9918	0.0043	0.991
50	100	0.01	0.8	0.0028	0.9913	0.047	0.9893
60	100	0.01	0.8	0.029	0.9911	0.0048	0.9925
70	100	0.01	0.8	0.0028	0.9928	0.0048	0.9898
80	100	0.01	0.8	0.0028	0.9919	0.0047	0.992
90	100	0.01	0.8	0.0028	0.9921	0.0046	0.9918
100	100	0.01	0.8	0.0029	0.9914	0.0052	0.9896

FIG. 30 – Résultat des comparaisons entre différents neurones dans la couche cachée

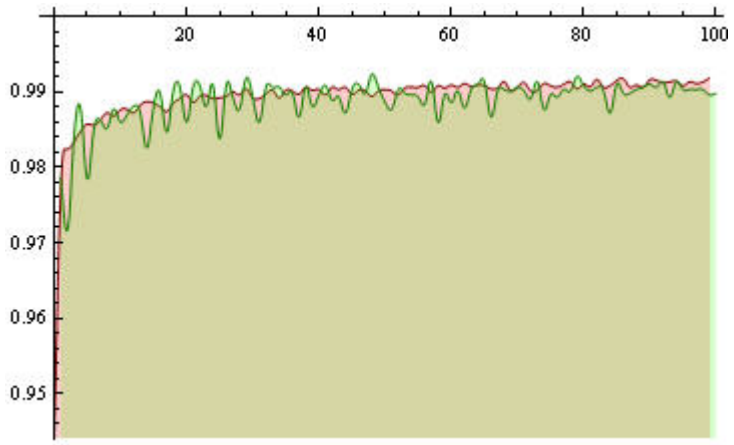

FIG. 31 – Graphique de l'évolution du taux de reconnaissance en fonction du nombre d'itérations avec 70 neurones dans la couche cachée

4.5.2 Optimisation de la valeur des paramètres η et μ

Une fois que le nombre optimal de neurones dans la couche cachée est trouvé, il est possible d'améliorer le pourcentage de détection en modifiant la valeur du taux d'apprentissage η. Une série de tests est effectué avec 70 neurones dans la couches cachées en faisant varier η de 0.01 à 0.9. La figure 32 montre que la valeur 0.3 semble être la plus optimale : elle permet d'améliorer le pourcentage de bonne détection de l'ensemble d'entraînement à 99.37 % et celui de l'ensemble de validation à 99.19%.

La figure 33 décrit l'évolution du taux de reconnaissance de l'ensemble d'entraînement en rouge et celui de l'ensemble de validation en vert avec un taux d'apprentissage de 0.3.

Une fois trouvée la valeur optimale qui est de 0.3 pour le taux d'apprentissage η, le moment de rétropropagation μ subit une variation entre 0.1 et 1 pour essayer de faire augmenter encore le taux de reconnaissance des chiffres. La figure (Fig. 34) donne la solution qui fournit le meilleur résultat ; celui-ci a une valeur de 0.6.

La figure 35 décrit l'évolution du taux de reconnaissance de l'ensemble d'entraînement en rouge et celui de l'ensemble de validation en vert avec un taux d'apprentissage de 0.3 et un moment de rétropropagation de 0.6..

Hidden	Epoch	nu	mu	train error	train rate	valid error	valid rate
70	100	0.01	0.8	0.0028	0.9928	0.0048	0.9898
70	100	0.02	0.8	0.0029	0.9919	0.006	0.9906
70	100	0.03	0.8	0.0028	0.9909	0.0052	0.9915
70	100	0.04	0.8	0.0029	0.9911	0.047	0.9923
70	100	0.05	0.8	0.028	0.9916	0.0048	0.9906
70	100	0.2	0.8	0.003	0.9917	0.0049	0.9935
70	100	0.3	0.8	0.028	0.9937	0.0047	0.9918
70	100	0.4	0.8	0.003	0.9906	0.0054	0.9884
70	100	0.5	0.8	0.0028	0.9917	0.061	0.9906
70	100	0.6	0.8	0.0028	0.992	0.0065	0.9908
70	100	0.7	0.8	0.028	0.9915	0.0047	0.9901
70	100	0.8	0.8	0.0029	0.992	0.0061	0.9906
70	100	0.9	0.8	0.0028	0.9919	0.008	0.9567

FIG. 32 – Résultat des comparaisons en faisant varier η dans le réseau de neurones

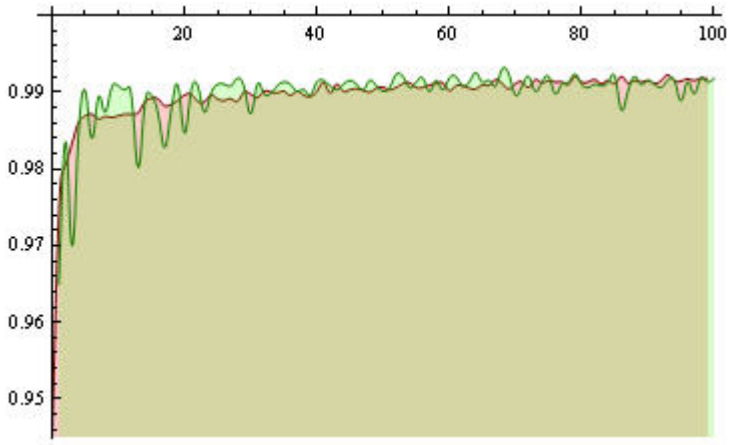

FIG. 33 – Graphique de l'évolution du taux de reconnaissance en fonction du nombre d'itérations avec 70 neurones dans la couche cachée et $\eta = 0.3$

Hidden	Epoch	nu	mu	train error	train rate	valid error	valid rate
70	100	0.3	0.1	0.0029	0.9911	0.0051	0.9893
70	100	0.3	0.2	0.0028	0.9911	0.0048	0.9908
70	100	0.3	0.3	0.028	0.9908	0.0057	0.9893
70	100	0.3	0.4	0.0028	0.9918	0.0055	0.9864
70	100	0.3	0.5	0.0029	0.991	0.0054	0.991
70	100	0.3	0.6	0.0029	0.9943	0.0052	0.992
70	100	0.3	0.7	0.0028	0.992	0.0063	0.9891
70	100	0.3	0.8	0.0028	0.9937	0.0047	0.9918
70	100	0.3	0.9	0.0027	0.992	0.0053	0.9918
70	100	0.3	1	0.027	0.9925	0.0043	0.9918

FIG. 34 – Résultat des comparaisons en faisant varier μ dans le réseau de neurones

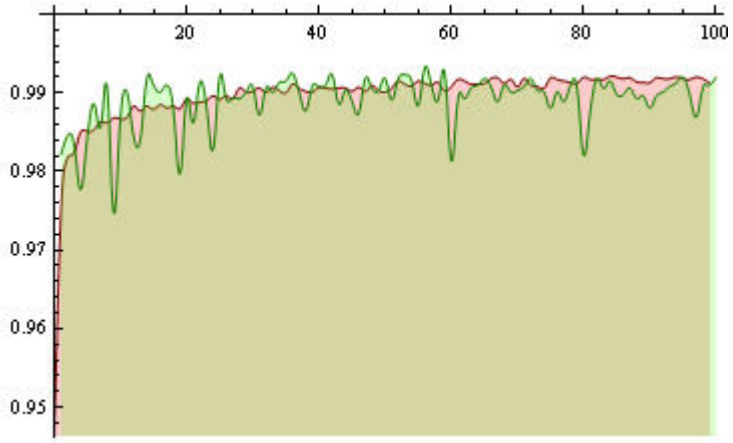

FIG. 35 – Graphique de l'évolution du taux de reconnaissance en fonction du nombre d'itérations avec 70 neurones dans la couche cachée, $\eta = 0.3$ et μ=0.6

5 Résolution du sudoku

Le sudoku est un jeu en forme de grille composée de 81 cases et inspiré du carré latin ainsi que du problème des 36 officiers du mathématicien suisse Leonhard Euler.

Le but du jeu est de remplir cette grille avec une série de chiffres tous différents, qui ne se trouvent jamais plus d'une fois sur une même ligne, dans une même colonne ou dans une même sous-grille.

Les symboles sont des chiffres allant de 1 à 9, les sous-grilles étant alors des carrés de 3 x 3 (Fig. 36).

La résolution des grilles est effectuée avec la méthode de résolution *brute force* qui teste toutes les possibilités de résolution de la grille, pour autant qu'il y en ait une. Cette méthode n'est pas la plus judicieuse mais donne un bon résultat avec un temps de calcul inférieur à une demie-seconde pour les grilles les plus difficiles. [8].

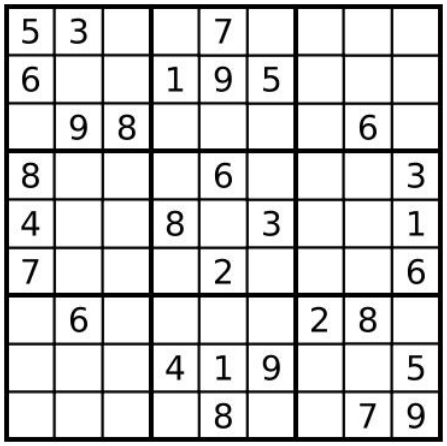

FIG. 36 – Grille de sudoku classique avec ses 9 sous-grilles 3 X 3

6 Portabilité

Il est possible d'effectuer la reconnaissance et la résolution de grilles de sudoku à partir de photos prises avec des téléphones portables. A présent il reste à déterminer comment utiliser pratiquement ce système pour le proposer à des utilisateurs. Trois utilisations possibles peuvent être envisagées :

- Utiliser un système de client serveur.
- Utiliser une application J2EE pour téléphone portable.
- Utiliser un système hybride ayant une partie sur un téléphone et l'autre sur un serveur.

Il faut remarquer que le temps de calcul pour arriver à une résolution complète, c'est à dire, de la photo à la résolution du sudoku prend environ 25 secondes sur un ordinateur possédant un processeur de 2.2 Ghz avec une mémoire de 2Gb de RAM. Si une comparaison de puissance de calcul est effectuée entre des modèles de téléphones portables les plus récents et les plus puissants qui existent actuellement sur le marché tels que l'iPhone, le Sony Ericson X1, etc... avec un ordinateur de bureau, nous pouvons remarquer que cette puissance est divisée par 4 (Fig. 37). La plupart des téléphones normaux comme la série 60 chez Nokia possède un processeur ayant une puissance de 100 Mhz, et une mémoire vive de 3.5 Mb, ce qui signifie que la puissance de calcul est divisée par 20 et la mémoire vive est diminuée de plus de 500 fois celle d'un ordinateur.

Sun nous propose une version allégée de J2SE, adaptée aux appareils de faible puissance appelée J2ME. Il est alors très certainement possible d'adapter le code utilisé pour résoudre le problème défini dans ce rapport en J2ME. Cependant la puissance de calcul est tellement importante pour le prétraitement des images que l'utilisateur devrait attendre 3 minutes pour obtenir le résultat de son sudoku sur un téléphone portable puissant et plus de 8 minutes sur un

téléphone portable normal. Il est alors très peu recommandé de fournir aux utilisateurs un système aussi lent sur leurs téléphones portables. Il faut aussi souligner que le temps d'utilisation de la batterie de ces téléphones sera nettement diminué après un processus utilisant 100% des ressources du téléphone portable pendant près de 10 minutes.

Il serait bien plus judicieux d'envoyer une image directement à un serveur, avec la technologie MMS par exemple, pour effectuer toutes les étapes sur le serveur afin qu'il renvoie la solution directement sur le téléphone portable de l'utilisateur. De plus cette solution est commercialement plus intéressante vu qu'elle génèrerait du traffic sur le réseau de l'opérateur téléphonique qui fournirait ce service.

La dernière solution est une solution hybride qui comprendrait un module sur le téléphone et un serveur. L'image pourrait être cadrée et traitée sur le téléphone avant d'être envoyée au serveur, cela éviterait des envois d'images trop grandes ou mal cadrées. Cette option nécessiterait le téléchargement par l'utilisateur d'une application sur son téléphone.

Le choix de la méthode de distribution de ce système de résolution dépend du marché que l'on veut atteindre. Si la cible est des utilisateurs occasionnels qui désirent une solution rapide de leur sudoku, il n'est pas possible de leur imposer de télécharger une application. Cependant si une clientèle de fans de sudoku est ciblée, il est envisageable de leur faire télécharger une application sur leur téléphone portable, dans le but de l'utiliser souvent.

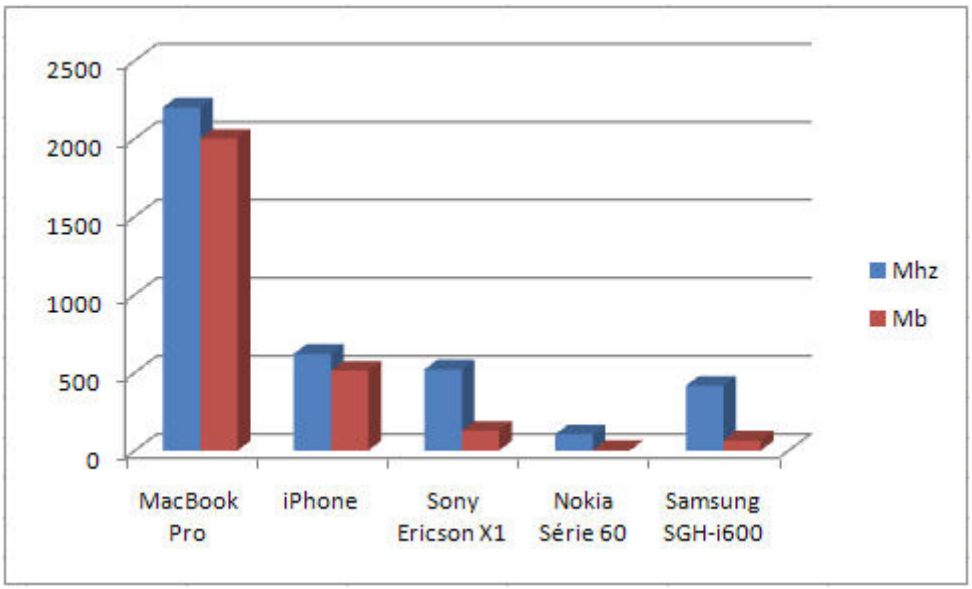

FIG. 37 – Comparaison de puissance entre téléphones portables et ordinateurs

7 Amélioration

Le système actuel est la première version qui donne des résultats concluants et satisfaisants. Il est certain qu'après avoir étudié toutes ces étapes, quelques améliorations seront envisageables pour le futur.

7.1 Détection des mauvaises images

Actuellement le système ne fait aucune vérification sur l'image qu'il doit traiter. Si ce système est proposé à des utilisateurs, il serait important d'effectuer les vérifications suivantes :

- Vérification que toute la grille figure sur la photo.
- Vérification que l'image est à l'endroit, en l'occurrence effectuer les rotations nécessaires.
- Vérifier le niveau de netteté de l'image.

7.2 Détection de la grille

Comme énoncé précédemment, la détection de la grille se fait avec les composantes connexes qui isolent la grille et avec la mesure des pics des projections des pixels sur les axes des X et Y. Il serait possible d'envisager une détection uniquement avec les composantes connexes. Avec le traitement de l'image, il est possible d'obtenir une image binarisée de bonne qualité. Il serait alors possible de détecter directement les 81 cases de la grille avec des composantes connexes. Ainsi on pourrait reconstruire une grille de sudoku même extrêmement déformée en détectant toutes composantes connexes de l'image et en déduire avec un modèle l'emplacement des cases de la grille sur l'image.

7.3 Réseau de neurones

Le réseau de neurones donne de très bons résultats en s'entraînant seulement avec les échantillons des cases. Il serait optimal d'inclure les règles de base du sudoku dans la reconnaissance de façon telle qu'un même chiffre ne se trouve jamais plus d'une fois sur une même ligne, dans une même colonne ou dans une même sous-grille. Cet ajout permettrait de rendre le réseau de neurones encore plus performant.

7.4 Résolution améliorée

Le système actuel permet seulement de résoudre des grilles vierges. Cette résolution pourrait être améliorée en proposant une méthode de résolution sur une grille déjà commencée à la main par un utilisateur. Il serait alors possible de proposer un mode qui montrerait si des erreurs sont commises sans montrer la résolution de la grille en entier.

8 Conclusion

Le but de ce projet était d'obtenir un système permettant la résolution de grilles de sudoku à partir d'une photo prise par un téléphone portable.

Différentes méthodologies de traitement d'images ont été évaluées afin de permettre une bonne détection de la grille. Pour notre problématique, la meilleure combinaison de traitement reprend les algorithmes suivants :

- Redimensionnement de l'image.
- Transformation de l'image en niveau de gris.
- Normalisation d'histogramme.
- Binarisation de l'image avec l'algorithme de Sauvola.

Par la suite, différentes méthodes et algorithmes ont été testés afin d'optimiser la détection de la grille. L'algorithme fournissant le meilleur résultat en temps de calcul ainsi qu'en pourcentage de grilles détectées est la méthode des composantes connexes. Cette méthode permet d'isoler la grille du reste de l'image. La détection de la grille s'effectue avec l'aide des projections des pixels noirs de l'image sur les axes X et Y.

La reconnaissance des chiffres du sudoku a été réalisée avec un réseau de neurones artificiels dont les différents paramètres (nombres de neurones, taux d'apprentissage, moment de rétropropagation) ont été optimisés afin de maximiser le taux de reconnaissance sur la base de données.

La résolution du sudoku à partir du résultat de la reconnaissance a été implémenté par un *solver* développé au DIUF dans le cadre d'un autre projet.

Finalement, une analyse des différentes stratégies pour une implémentation sur un téléphone portable a été réalisée. La conclusion de cette analyse est qu'une intégration complète de la solution sur téléphone mobile est actuellement difficile à cause de la quantité de calculs à réaliser. Une stratégie client-serveur serait plus appropriée avec la prise de la photo réalisée sur le téléphone ainsi qu'un éventuel traitement de l'image et transfert de la photo, ou de ses caractéristiques extraites, sur un serveur via une connexion GPRS, UMTS, WIFI ou autre.

Malgré le fait que les téléphones portables actuels ne soient pas conçus pour prendre des photos de près, ce travail de Master démontre qu'en travaillant sur les images et en y appliquant des traitements, des résultats très concluants peuvent être obtenus. D'autres approches peuvent être possibles mais celle qui a été choisie semble être pour l'instant la meilleure.
Ce travail a traité de nombreux sujets dans différents domaines comme les déformations des images prises avec des appareils-photos, le traitement d'images, la création d'un réseau de neurones pour la reconnaissance des chiffres et finalement la résolution de la grille d'un sudoku.

Remerciements

M. Jean Hennebert, maître assistant, Université de Fribourg.

M. Robert VanKommer, Swisscom Innovations.

M. Jean-Luc Bloechle, assistant et doctorant, Université de Fribourg.

Références

[1] P. Clark and M. Mirmehdi. Estimating the orientation and recovery of text planes in a single image.

[2] Jian Liang David Doermann and Huiping Li. Process in camera-based document image analysis. *Proceedings of the Seventh Internationnal Conference on Document Analysis and ¿Recognition (ICDAR 2003)*, 2003.

[3] David Doermann Huiping Li Jian Liang. Camera-based analysis of text and documents : a survey. *International journal on document analysis and recognition*, pages 88–200, 2005.

[4] Huiping Li Jian Liang, David Doermann. Camera-based analysis of text and documents : a survey. *International Journal on Document Analysis and Recognition*, 2005.

[5] Y.G. Chung W.P. Yu K.S. Bae, K.K. Kim. Character recognition system for cellular phone with camera. *Proceedings of the 29th Annual Internationnal Computer and Application Conference (COMPSAC'05)*, 2005.

[6] C.M. Lee and A. Kankanhalli. Automatic extraction of characters in complex images. *International Journal of Pattern Recognition Artificial Intelligence*, pages 67–82, 1995.

[7] W Niblack. An introduction to digital image processing. *Englewood Cliffs, NJ : Prentice Hall.*, 1986.

[8] Juillerat Nicolas. http ://diuf.unifr.ch/people/juillera/sudoku/sudoku.html. 2007.

[9] Bui Truong Minh Marius Bulacu Nobuo Ezaki, Kimiyasu Kiyota and Lambert Schomaker. Improved text-detection methods for camera-based text reading system for blind persons.

[10] N.Otsu. A threshold selection method from grey scale histogram. *IEEE Trans. on Syst. Man and Cyber*, pages 62–66, 1979.

[11] Peter E. Hart Richard O. Duda and David G. Stork. Pattern classication, second edition. *Library of Congress*, 2001.

[12] J. Sauvola and M. Pietikainen. Adaptive document image binarization. *Pattern Recognition 33*, pages 225–236, 2000.

[13] Shinichiro Omachi Koichi Kise Seichi Uchida, Masakazu Iwamura. Ocr fonts revisited for camera-bases character recognition. *The 18th Internationnal Conference on Pattern Recognition (ICPR'06)*, 2006.

[14] Seiichi Uchida Koichi Kise Shinichiro Omachi, Masakazu Iwamura. Affine invariation information embedment for accurate camera-based character recognition. *The 18th Internationnal Conference on Pattern Recognition (ICPR'06)*, 2006.

[15] Chahine A David L Zandifar A, Duraiswami R. A video based interface to textual information for the visually impared. *Rev. Mod. Phys.*, page 325, 2002.