FACULTE DES SCIENCES - Année 2018-2019

DEPARTEMENT INFORMATIQUE

Rapport de projet TER
Projet Informatique HLIN405

Projet Sudoku en Réalité Augmentée

Encadrant:
Boudet Vincent

Etudiants:
Simione Jérémy, Henriksen Leif

Remerciements

Je tiens à remercier toutes les personnes qui m'ont aidé à rédiger cet article, Namrod pour la partie bibliographie, Francis Walter, pour ses conseils ainsi que les personnes ayant participé à la correction de ce document.

Contents

Introduction

Dans le cadre de L'UE Projet de Programmation , nous avons développé une application permettant à un utilisateur de prendre une grille de Sudoku en photo pour pouvoir la résoudre directement ou alors jouer.

Notre groupe est composé de trois personnes, Jérémy SIMIONE, Leif HENRIKSEN et Thomas BESSON, et nous sommes encadrés par M. Vincent BOUDET.La réalisation du projet s'est déroulée sur une période de 13 semaines de fin janvier à fin avril.

Motivations du Projet

L'utilisation de l'intelligence artificielle pour le traitement d'image et la réalisation d'une application Android pour l'implémenter est un sujet qui nous a directement intéressé, ce sont des systèmes qui se développent de plus en plus aujourd'hui car ils sont très efficaces en reconnaissant des chiffres ou tout type d'objets.

Ces systèmes sont très developpés chez Google, Amazon ou Microsoft pour des voitures autonomes, des magasins autonomes ou toute autre système autonome.

Objectifs du projet et cahier des charges

L'objectif est de permettre à l' utilisateur de prendre une photo d'une grille de Sudoku afin qu'il puisse obtenir la solution de la grille, le second objectif est qu'il puisse jouer sur l'application. Il existe déjà des applications Android capables de réaliser cet objectif, mais la plupart ont du mal a obtenit un résultat correct.

Voici le cahier des charges que nous avons crée pour notre application :

Création d'un algorithme de résolution
Création d'une classe Sudoku qui va représenter notre grille de sudoku mais aussi toutes les méthodes qui vont nous permettre de récuperer la grille ainsi que l'algorithme de résolution d'une grille.

Création de l'interface graphique de l'application
L'interface graphique nous permettra de utiliser l'application en appuyant sur des boutons. Nous allons créer une classe Grille qui prend une grille en entrée qui vient de l'algorithme de résolution ou de l'analyse d'image et qui va l'afficher sur l'application. Cette classe Grille nous permettra d'utiliser notre grille.

Mise en place d'un accès a la caméra
Création d'une classe dédiée à la caméra pour que l'utilisateur puisse prendre des photos d'une grille. Cette partie nous servira a envoyer l'image a notre derniere partie.

Mise en place d'un système d'analyse d'images
Création de différentes classes qui vont nous servir à traiter l'image en utilisant la bibliothèque OpenCV et ainsi pouvoir procéder à une reconnaissance des chiffres et l'afficher sur la grille de l'application dans le cas ou l'utilisateur voudrait résoudre une grille ou jouer.

Création du jeu
Création d'une partie jouable en utilisant l'interface graphique de l'application ainsi que la classe Sudoku.

Rassemblement des différentes parties
S'assurer de la compatiblite de tous les classes pour avoir le comportement dessire, et faire les modifications necessaires à la interface graphique pour pouvoir accerder a tous les fonctionalites de l'application.

1 Organisation du projet

1.1 Organisation du travail

Pour le développement de notre application Sudoku , nous avons décidé de travailler chacun de notre côté et de temps en temps ensemble suivant la difficulté des choses à réaliser.Pour que le projet nous apporte a tous des connaissances dans les différents domaines auquel il touche nous avons essayé de repartir les tâches de sorte a ce que chaque membre du groupe ai vu chaque domaine (android,java,openCV).

Afin d'être les plus efficace et d'avancer le plus rapidement possible nous nous sommes réunis quotidiennement. Durant les jours de la semaine, nous nous sommes vus souvent afin de connaître l'avancée de chacun dans le projet,faire le point sur l'avancement du projet, définir de nouveaux objectifs et de les réaliser.

A chaque étape réalisée nous avons postés sur un depot Github créé pour le projet chaque nouvelle partie afin que tout chaque memebre puisse s'informer et voir. A chaque étape importante nous nous sommes réunis avec notre encadrant M. Boudet afin de faire le point sur l'état d'avancement de l'application.

1.2 Répartition du travail dans le temps

Nous avons découpé cette période de travail en plusieurs phases.

1. **Préparation du projet**
   Nous avons réalisé le cahier des charges de l'application, choisi les outils de travail et les principales technologies utilisées. Nous avons fait une première version du diagramme de répartition des tâches dans le temps, et une première modélisation de l'architecture de l'application.

2. **Développement du projet**
   Nous avons implanté les fonctionnalités de l'application en raffinant la modélisation au fur et à mesure. Pour chaque module implanté, nous nous sommes efforcés d'écrire des tests afin de s'assurer de leur bon fonctionnement.

3. **Finalisation du projet**
   Cette phase a consisté en la correction de bogues afin d'obtenir une version suffisamment stable pour pouvoir être présentée en vue de la soutenance et du rendu du projet T.E.R.

1.3 Outils de travail collaboratif

Nous avons choisi d'utiliser Github qui permet la gestion des versions du projet et facilite la collaboration a distance.

Enfin, pour éditer le code du projet , nous nous sommes servis d'Android Studio.

Il etait en effet plus facile de commencer sur cet éditeur car il ne fonctionne pas de la meme facon que les autres éditeurs et notre code source final passe obligatoirement par cet IDE. Nous avons aussi créer un diagramme de gantt afin de planifier les tâches pour avoir des dates limites pour chaque partie ce qui nous a permis de réaliser le projet dans son ensemble et dans les temps.

Pour rédiger les différents documents, y compris ce rapport, nous avons utilisé LaTeXpour sa capacité à produire des documents de bonne qualité.

2 Conception du Sudoku

Nous avons décidé de faire la conception de ce projet en 5 parties, analyse d'image, interface graphique et interaction, le jeu, algorithme de résolution, et rassemblement des parties. Nous avons essayé d'avoir une intersection vide entre les 5 parties, avec le but de pouvoir travailler parallèlement, donc à part la partie rassemblement et interface graphique avec jeu, les différents parties sont indépendants.

Tout d'abord nous allons vous expliquer l'idée générale de notre application, comment elle fonctionne et ses objectifs, puis nous allons décrire avec plus de détails les 5 parties mentionnées précédemment dans l'ordre suivant, premièrement l'analyse d'image, deuxièmement l'interface graphique et l'interaction, troisièmement l'algorithme de résolution, quatrièmement le jeu, et cinquièmement le rassemblement des parties.

2.1 Général

Notre application à pour but de prendre en photo une grille de sudoku, analyser cette image, extraire les valeurs des cases et les afficher dans une grille, une fois que les valeurs sont dans l'application, on peut choisir de jouer ou de résoudre la grille en utilisant l'algorithme de résolution.

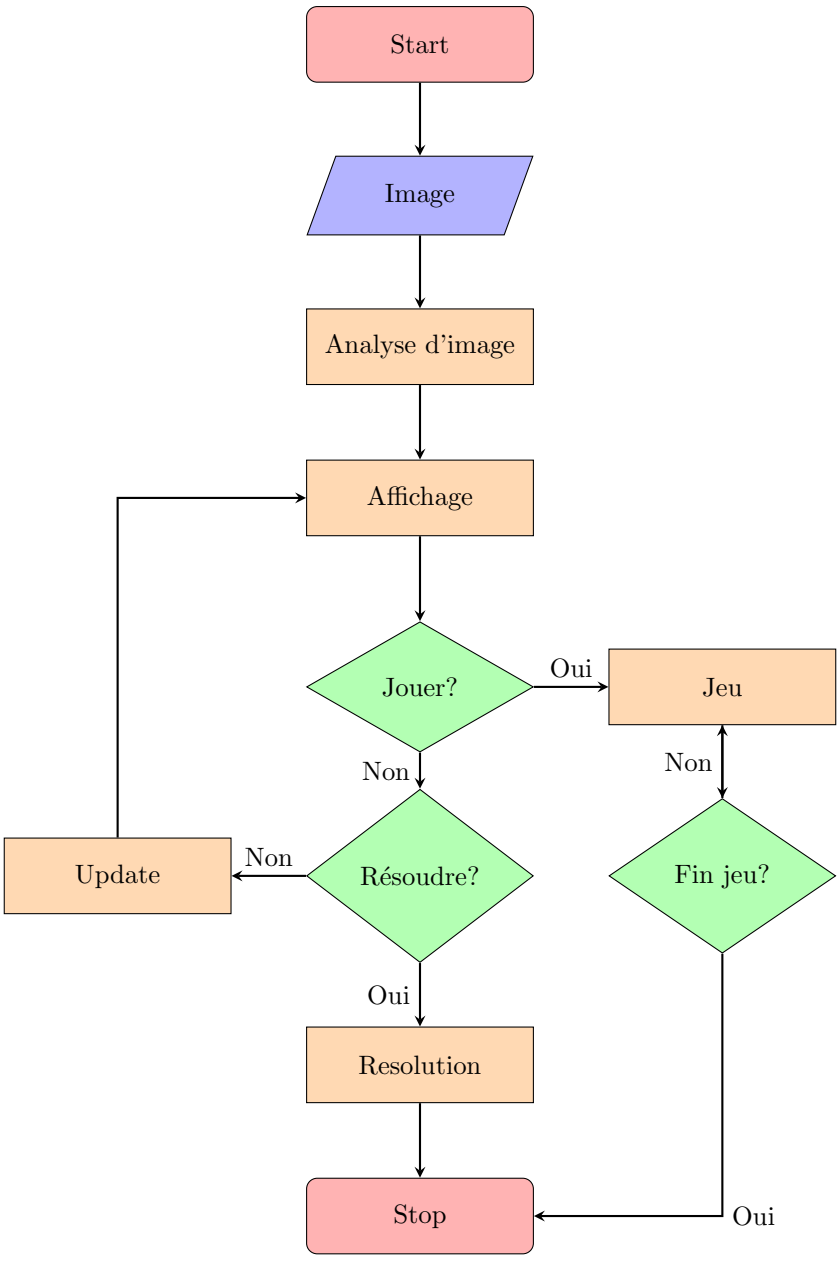

Figure 1: Schéma du fonctionnement général de l'application

2.2 Analyse d'image

2.3 Interface graphique et interaction

L'interface graphique est divisé en deux components principales, d'abord la grille de 81 cases, et après, les boutons nécessaires à l'utilisation de l'application. La deuxième partie est simplement une interface de menu classique, différentes boutons qui donneront accès aux autres parties de l'application. La première partie est chargée de l'affichage des données qui vient de l'analyse d'image ou de l'algorithme de résolution, et aussi est charge de permettre à l'utilisateur de modifier les valeurs des casses, pour qu'il puisse modifier la grille s'il y a eu un erreur avec l'analyse d'image ou s'il a choisi de jouer. Donc elle est beaucoup plus compliqué et nous allons l'expliquer plus en détaille.

Nous avons créé 4 classes pour mettre en place la grille,

- **EditeurGrille**
  La classe EditeurGrille est chargée de gérer les valeurs qui viennent de l'analyse d'image et l'algorithme de résolution, aussi du control des boutons pour la modification des valeurs.
- **Grille**
  La classe Grille contient tout les méthodes nécessaires pour faire la mise à jour de la grille, Grille lit les couleurs des casses, le nombre de chaque casse, et les affiche correctement.
- **Cases**
  La classe Cases contient 81 objets Case, et elle est chargée de les modifier tous ensemble et les gérer.
- **Case**
  La classe Case représente une casse dans la grille, elle contient plusieurs attributs utiles pour l'éditeur et pour le jeu, principalement une Case contient une valeur et une couleur(blanc, bleu ou rouge), le couleur blanc représente une casse modifiable, bleu une casse non modifiable (si on n'est pas en mode jeu on peut toujour modifier une casse bleu), et rouge une casse avec une mauvaise valeur (utilisée seulement dans le jeu) . Case contient aussi autres attributs et méthodes mais ils seront expliqué avec plus de détaille dans la partie Jeu.

2.4 Algorithme de résolution

Jeu

Pour le jeu nous avons créé la classe Jouer qui hérite de la classe EditeurGrille, le jeu fait la même chose que EditeurGrille, mais utilise des attributs dans la classe Case pour la logique de jeu. La classe Case contient deux attributs importants pour le jeu, estModifiable et valeurReponse, quand nous essayons de changer la valeur d'une casse d'abord nous vérifions si elle est modifiable, après nous confirmons si nous avons mis la bonne valeur, si c'était le cas nous la transformons en non modifiable, nous changeons la valeur et la couleur en bleu et nous enlevons 1 au compteur de casses vides, sinon nous changeons la couleur en rouge et ajoutons 1 au compteur d'erreurs. Le jeu finit quand le nombre de cases vides est equal a 0.

Rassemblement

Pour le rassemblement nous nous avons assurer que les autres parties étaient le plus indépendantes possibles, donc pour les rassembler nous avons utilisé les outils de notre système d'exploitation, Android, pour qu'elles puissent communiquer entre elles, et grâce à leur indépendance nous avons évité centaines de bugs. L'analyse d'image envoi de l'information à l'éditeur de grille, puis si le joueur souhaite la résolution, l'algorithme de résolution lui donne la réponse, sinon l'éditeur de grille envoi l'information de la grille à la classe Jouer, et le joueur peut jouer.

3 Implémentation du Sudoku

3.1 Création de l'algorithme de résolution

La première partie consistait a créer une classe java de sudoku pour pouvoir implémenter la résolution; la solution la plus simple était de choisir une structure de données de type tableau à deux dimenions.

La deuxième étape consitait a créer des fonctions auxiliaires qui nous servirait pour la résolution de la grille entière; nous avons donc créer des fonctions qui testent si une valeur est absente d'un bloc, d'une colonne ou d'une ligne de la grille ; voici un exemple de l'une de ces fonctions:

Algorithme 1 : absentSurColonne(Entier valeur,Grille g,Entier j)

```
Données : Entier valeur,Grille grille,Entier j
Résultat : Renvoi vrai si la valeur n'est pas dans la colonne, faux sinon
pour i allant de 0 à 9 faire
    si grille[i][j]=k alors
        retourner Faux
    fin
    retourner Vrai
fin
```

La troisième étape consistait à implémenter le backtrack[1].Cette fonction doit prendre une grille en entrée la résoudre et nous informer de l'état du réultat en nous renvoyant un booléen. Il fallait donc vérifier dans la descente récursive en énumérerant tous les chiffres possibles pour observer si nous arrivions à un résultat correct ou un blocage tout cela en remplissant la grille dans la descente et si nous arrivions à un blocage nous reinitialisions la case correspondante à zéro.
Voici l'algorithme en question :

Algorithme 2 : estValide(Grille grille,Entier position)

```
Données : Grille grille,Entier position
Résultat : Renvoi vrai si la grille a été résolue, renvoi faux sinon
si position=9*9 alors
    retourner Vrai
fin
i ⟵ position ÷ 9 j ⟵ position%9
si grille[i][j]!=0 alors
    retourner estValide(grille,postion+1);
    pour k allant de 1 à 9 faire
        si absentSurLigne(k,grille,i) et absentSurColonne(k,grille,j) et absentSurBloc(k,grille,i,j)
         alors
            grille[i][j] ⟵ k;
            si estValide(grille,position+1) alors
                retourner Vrai
            fin
        fin
        grille[i][j] ⟵ 0
        retourner Faux
    fin
fin
```

[1] Aussi nommé le retour sur trace en français

3.2 Mise en place d'une caméra

L'objectif de cette partie est de fournir à l'utilisateur un accès a la caméra afin qu'il puisse prendre des photos des grilles qu'il souhaite résoudre ou jouer.

Dans cette classe nous nous sommes servis de l'API caméra déjà existante sur le téléphone au lieu de faire une nouvelle API.

Nous avons donc créer un accès a la caméra par notre application en gérant en premier lieu les permissions de l'application; c'est a dire que l'application puisse écrire des fichiers sur le stockage externe du téléphone afin de sauvegarder les images capturées sur le téléphone mais aussi qu'elle ai une permission d'ouverture de l'application caméra déjà présente sur le terminal.

Il a ensuite fallu créer un bouton qui accédait a cette caméra et une vue de l'image résultante afin de vérifer que l'utilisateur puisse voir si la photo lui convenait.

L'implémentation de cette partie n'a pas présenté de complexité technique particulière mis a part la création du chemin de l'image qui s'est avérée compliqué.

3.3 Mise en place d'un système de reconnaissance des chiffres

Pour réaliser le système de reconnaisance des chiffres, nous avons du utiliser la bibliothèque graphique OpenCV. La plupart des problèmes que nous pouvons rencontrer avec cette bibliothèque ne sont très bien documentés car c'est une bibliothèque technique et utilisé surtout par des gens qualifiés.

Nous avons opté pour une implémentation permettant d'afficher sur une grille la reconnaissance d'une image provenant d'un traitement.Le but est que du moment ou l'image a été traitée elle est directement envoyé a l'IA pour qu'elle soit ensuite directement affichée sur l'application.

Lorsque l'utilisateur va sélectionner l'image elle sera directement envoyé au programme de traitement de l'image qui lui-même envera le résultat a l'IA qui essaiera de reconnaitre les chiffres puis d'envoyer un résultat a la classe Sudoku qui l'affichera par la suite.

Pour cette partie nous avons décidés de nous faciliter la tâche en utilisant l'IDE eclipse car la gestion des chemins d'accès des images[2] était plus aiséé et ainsi pouvoir observer les traitements résultants afin d'obtenir une image lisible par l'ordinateur.

Pour implémenter ce traitement il y avait un cahier des charges à suivre :

Premièrement il fallait détecter la grille en appliquant une série de traitements à l'image a savoir:

- **Convertir notre image de base avec des teintes grises**
  Pour cela il fallait créer deux matrices qui sont les conteneurs de l'image; une pour l'image source et une pour l'image transformée ensuite OpenCV disposait de méthodes permettant la conversion directe voici un exemple de traitement:

[2]En effet lors de la création d'une application Android les chemin d'accès aux images doivent provenir exclusivement du téléphone.

Données : Matrice source, Matrice destination, String chemin
Résultat : Transforme l'image source en gris
//On charge la librairie
$System.loadLibrary(CORE.NATIVE_LIBRARY_NAME)$
//On lit l'image
$source = Imgcodecs.imread(chemin)$;
//On transforme l'image et on l'enregistre dans la matrice de destination
$Imgproc.cvtColor(soure, destination, Imgproc.COLOR_RGB2GRAY)$;
//On écrit l'image a un nouveau chemin
$Imgcodecs.imwrite(nouveauchemin, destination)$

- **Appliquer un flou gaussien afin de réduire le bruit et les détails de l'image**
  Pour cette étape nous utilisons la même manière de procéder que la conversion en gris avec nos deux matrices et nos méthodes pour les images : Imgcodecs et Imgproc.Le flou gaussien va agir sur chaque pixel en essayer d'harmoniser l'image au maximum.

- **Appliquer un seuillage d'image c'est a dire convertir notre résultat précedant en noir et blanc**
  En vérifiant bien que l'image que nous prenions soit l'image qui a subi déjà deux transformations. C'est le dernier traitement avant la détection de la grille et des lignes.

- **Trouver les bordures de la grille**

- **Détecter les lignes de la grille**

Il fallait créer ensuite un système d'intelligence artificielle qui allait essayer de comparer chaque chiffre de la grille avec des ressources et renvoyer une grille en résultat.

La derniere partie consistait à lier les trois classes c'est a dire notre classe Sudoku, celle qui manipulait l'image et l'intelligence artificielle et à adapter notre code afin qu'il soit effectif sur un terminal android.

Chapitre 4

4 Bilan et difficultés rencontrées

4.1 Bilan de l'avancement du projet

Nous avons terminé l'implémentation de l'application. Pour tester l'application nous l'avons installé sur nos smartphones et essayé sur différentes images que ce soit sur une photo ou une image provenant du web le résultat est plutôt satisfaisant, le seul bémol est la rapdité du programme qui dépend vraiment du terminal d'execution car OpenCV utilise beaucoup de ressources pour fonctionner.

Nous avons également terminé la création de la partie jeu et crée une classe qui permet de sauvegarder la partie en cours mais aussi une classe qui permet de dire si un chiffre est cohérent.

Actuellement notre version de l'application permet à l'utilisateur de lancer une partie de sudoku et résoudre ou jouer une grille qu'il a pris en photo auparavant.

Enfin, nous sommes aujourd'hui en train de finaliser notre application afin qu'elle soit un peu plus estethique.

4.2 Difficultés rencontrées

Pour la partie "Mise en place d'un systeme de reconnaissance de chiffres" nous avons été confrontés pour l la première fois a une bibliotheques de traitement d'image et la gestion d'une intelligence artificielle.

Nous avons donc du nous former a cette bibliothèque et après de nombreux essais nous nous ne savions pas si notre système d'IA allait réussir à détecter notre image traitée c'est pourquoi après concertation avec notre encadrant nous avons décidé d'utiliser une classe déja existante nommée ImageManipulator qui traitait l'image comme nous avions besoin afin d'être sûr que le résultat serait correct et que l'image sera ainsi facilement lisible par l'IA.Nous sommes donc parvenus a surmonter cette difficulté en utilisant des classes deja existantes a savoir une classe qui nous a permis de manipuler l'image et une classe qui nous a servi d'intelligence artificielle.

Ensuite malgré la grande aide que nous ont apportés ces deux classes l'adaptation du code d'eclipse à Android Studio a été très difficile car il fallait déjà réussir a installer une version d'openCV sur l'IDE Android ce qui n'a pas été chose aisée, mais aussi une fois ceci fait, trouver une version qui compilait à l'appel de la bibliothèque et vérifier qu'openCV était bien fonctionnel.
Ensuite plus rien ne fonctionnait de la même façon il a donc fallu créer des classes qui nous ont permis de passer outre cette difficulté et pouvoir faire l'adaptation à Android Studio.

Cette partie a été de loin la plus longue et la plus dure.