

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, BESSON Thomas

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Objectifs du projet et cahier des charges . . . . .	2
<b>2</b>	<b>Organisation du projet</b>	<b>3</b>
2.1	Organisation du travail . . . . .	3
2.2	Répartition du travail dans le temps . . . . .	3
2.3	Outils de travail collaboratif . . . . .	4
<b>3</b>	<b>Conception</b>	<b>4</b>
3.1	General . . . . .	4
3.2	Analyse d'image . . . . .	6
3.3	Interface graphique et interaction . . . . .	6
3.4	Algorithme de resolution . . . . .	6
3.5	Jeu . . . . .	6
<b>4</b>	<b>Implémentation du Sudoku</b>	<b>7</b>
4.1	Création de l'algorithme de résolution . . . . .	7
4.2	Mise en place d'un système de reconnaissance des chiffres . . . . .	9
4.3	Création de Grille de jeu . . . . .	10
<b>5</b>	<b>Bilan et difficultés rencontrées</b>	<b>12</b>
5.1	Avancement du projet . . . . .	12
5.2	Difficultés rencontrées . . . . .	12

# 1 Introduction

## 1.1 Objectifs du projet et cahier des charges

L'objectif est de permettre à un 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 à obtenir un résultat correct.

Voici le cahier des charges que nous avons créé 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 récupérer 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 à 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 à envoyer l'image à notre dernière 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 où 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 compatibilité de tous les classes pour avoir le comportement désiré, et faire les modifications nécessaires à la interface graphique pour pouvoir accéder à tous les fonctionnalités de l'application.

## 2 Organisation du projet

### 2.1 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 à tous des connaissances dans les différents domaines auquel il touche nous avons essayé de répartir les tâches de sorte à ce que chaque membre du groupe ait 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 membre 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.

### 2.2 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.

## 2.3 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  $\text{\LaTeX}$  pour sa capacité à produire des documents de bonne qualité.

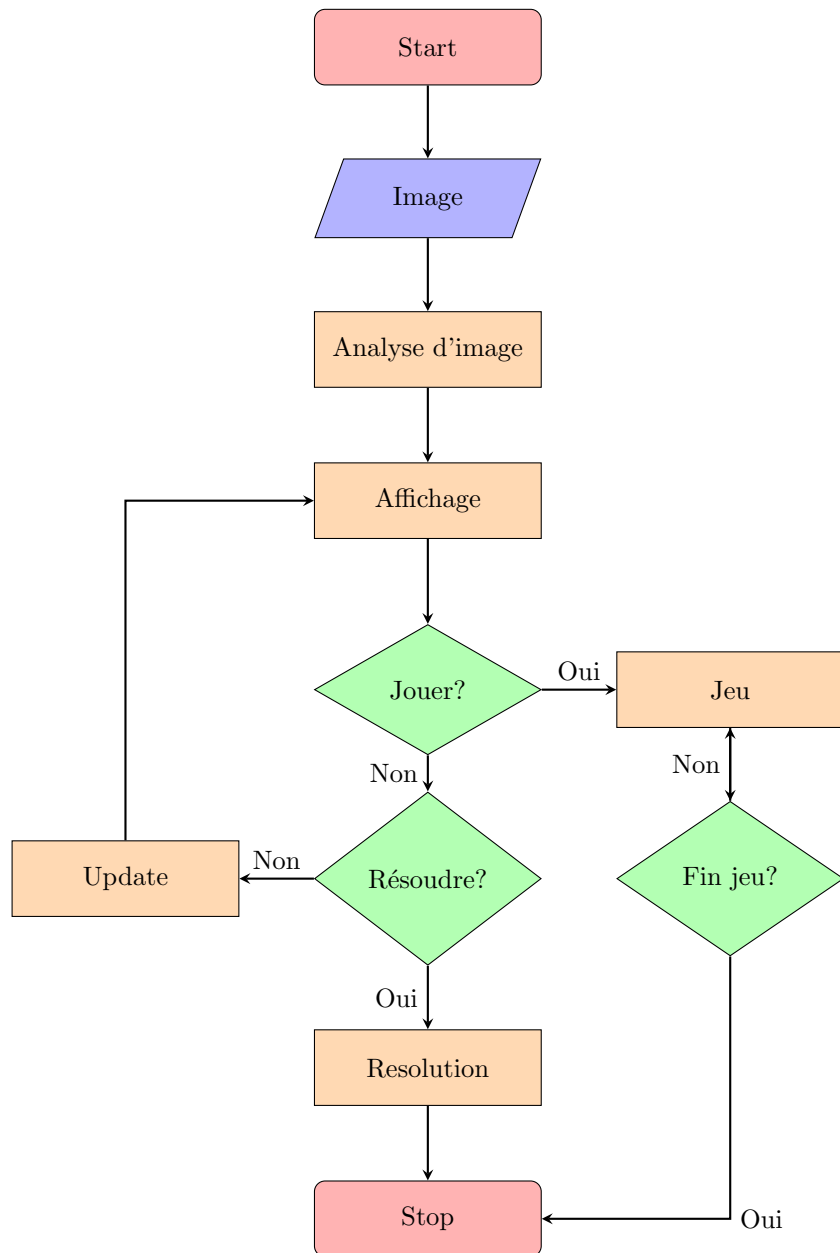
## 3 Conception

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.

### 3.1 General

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



## 3.2 Analyse d'image

## 3.3 Interface graphique et interaction

L'interface graphique est divisé en deux composants 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(bleu, blanc ou rouge), le couleur blanc représente une casse modifiable, bleu une casse non modifiable (si on n'est pas en mode jeu on peut toujours 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.

## 3.4 Algorithme de resolution

## 3.5 Jeu

Tout d'abord il a fallut créer différents niveau de jeu pour le sudoku. Un iveau facile, intermédiaire et un niveau difficile. Pour faire cela on a utilisé différentes strucutre. Une classe case pour chaque case du jeu et une classe Sudo9x9 permettant de pointer directement sur les cases via un tableau de 9 par 9. Cette classe Sudo9x9 permet de modifier les cases et de pouvoir résoudre un sudoku.

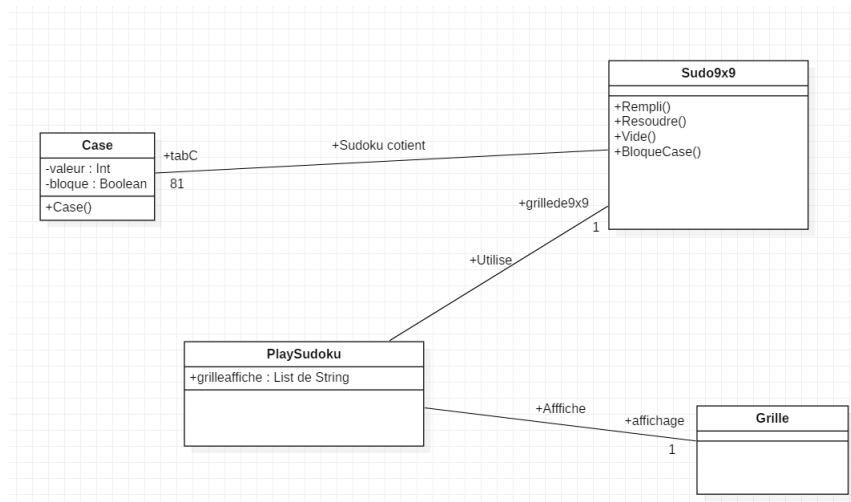


Figure 1: Diagramme UML pour jouer

Sur la figure 1 nous pouvons donc voir le graphe UML de la section jeux. Bien évidemment toutes les fonctions n'ont pas été représentées mais les plus importantes sont sur le graphe. Pour créer un niveau on va appeler une instance de PlaySudoku qui permet de créer les niveaux.

## 4 Implémentation du Sudoku

### 4.1 Création de l'algorithme de résolution

La première partie consistait à 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.

La deuxième étape consistait à créer des fonctions secondaires qui nous serviraient pour la résolution de la grille entière; nous avons donc créé 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<sup>1</sup>. Cette fonction doit prendre une grille en entrée la résoudre et nous informer de l'état du résultat en nous renvoyant un booléen. Il fallait donc vérifier dans la descente récursive en énumérant 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 réinitialisions 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**

$i \leftarrow \text{position} \div 9$   $j \leftarrow \text{position} \% 9$

**si** grille[i][j]!=0 **alors**

**retourner** estValide(grille,position+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]  $\leftarrow$  k;

**si** estValide(grille,position+1) **alors**

**retourner** Vrai

**fin**

**fin**

        grille[i][j]  $\leftarrow$  0

**retourner** Faux

**fin**

**fin**

---

<sup>1</sup> Aussi nommé le retour sur trace en français

## 4.2 Mise en place d'un système de reconnaissance des chiffres

Pour réaliser le système de reconnaissance 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 où l'image a été traitée elle est directement envoyée à l'IA pour qu'elle soit ensuite directement affichée sur l'application.

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

Pour cette partie nous avons décidé de nous faciliter la tâche en utilisant l'IDE eclipse car la gestion des chemins d'accès des images<sup>2</sup> était plus aisée 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 à savoir:

- Convertir notre image de base avec des teintes grises
- Appliquer un flou gaussien afin de réduire le bruit et les détails de l'image
- Appliquer un seuillage d'image c'est à dire convertir notre résultat précédent en noir et blanc
- Trouver les bordures de la grille
- Détecter les lignes de la grille

Voici ci-dessous des algorithmes de traitement de l'image utilisés par OpenCV:

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

Voici ci-dessous des algorithmes de traitement de l'image utilisés par OpenCV:

---

<sup>2</sup>En effet lors de la création d'une application Android les chemins d'accès aux images doivent provenir exclusivement du téléphone.

### 4.3 Création de Grille de jeu

(Tout les algorithmes présenter dans cette partie sont présent dans la classe Sudo9x9 qui a pour attribut un tableau de Case[9][9] appelé tabC).

Pour créer une grille jouable il faut d'abord savoir si la grille est résolvable. Pour ca on le fait en 3 étapes :

- Première étape :  
On va d'abord créer une grille donc un tableau de 9x9 (initialisé avec des cases vide ou plutôt à 0);
- Deuxième étape :  
On va donc remplir notre tableau avec 10 valeur placé au hasard et avec une valeur compris entre 1 et 9.

---

**Algorithme 3** : Rempli(Case[] T)

**Données** : Tableau à deux dimension de Case : T

**Résultat** : Rempli le tableau avec des valeurs au hasard

$i \leftarrow$  chiffre aléatoire compris entre 1 et 9;

$j \leftarrow$  chiffre aléatoire compris entre 1 et 9;

$c \leftarrow$  On cré un case;

**si**  $T(i,j) = 0$  **alors**

$T(i,j) \leftarrow c$ ;

**tant que** *Pas bon sur la ligne, la colonne et dans le bloc* **faire**

$c \leftarrow c.setValeur(\text{chiffre compris entre 1 et 9})$ ;

$T(i,j) \leftarrow c$ ;

**fin**

**fin**

rempli(tabC);

---

- Troisième étape :  
On résoud la grille sur notre tableau via notre algorithme de résolution, ensuite on vide la tableau selon un certain nombre et selon le niveau. Puis on bloque les cases qui reste.

---

**Algorithme 4** : Vide(int n)

**Données** : un entier n qui sera le nombre de case à vider

**Résultat** : vider tabC

$i \leftarrow 0$ ;

$j \leftarrow 0$ ;

**tant que**  $n$  *supérieur à 0* **faire**

$i \leftarrow$  prend un chiffre aléatoire entre 1 et 9;

$j \leftarrow$  prend un chiffre aléatoire entre 1 et 9;

**si** *valeur de tabC(i,j) != 0* **alors**

        valeur de tabC(i,j)  $\leftarrow 0$ ;

$n \leftarrow n-1$ ;

**fin**

**fin**

---

Notre algorithme final pour créer une grille de jeux est donc le suivant :

---

**Algorithme 5 :** Sudo9x9(int n)

---

**Données :** Un entier n pour savoir combien de case on va vider

**Résultat :** Créer notre grille de jeu selon le niveau

tabh  $\leftarrow$  tableau à deux dimension de 9 par 9;

**pour**  $j$  de 0 à 8 **faire**

**pour**  $k$  de 0 à 8 **faire**

        tabC(j,k)  $\leftarrow$  créer une case initialisé à 0;

**fin**

**fin**

**pour**  $i$  de 0 à 9 **faire**

    remplir tabC;

**fin**

tabh  $\leftarrow$  convertir tabC qui est un tableau de case en tableau d'entier  
pour résoudre notre grille;

resoudre(tabh,0);

tabC  $\leftarrow$  copie tabh dans tabC;

vide(n);

---

## 5 Bilan et difficultés rencontrées

### 5.1 Avancement du projet

### 5.2 Difficultés rencontrées

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

Nous avons donc dû nous former à cette bibliothèque et après de nombreux essais 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 déjà 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 à surmonter cette difficulté en utilisant des classes déjà existantes à 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 apportées ces deux classes l'adaptation du code d'eclipse à Android Studio a été très difficile car il fallait déjà réussir à 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.