Traitement d'image sous Android- Cahier des charges

Fabien Baldacci, Boris Mansencal, Anne Vialard

Complété par Abderrahmane Kerroum et Maxence Lévêque

Année Universitaire 2017/2018

Table des matières

Introduction

Ce document contient une description du travail fait dans le cadre du projet de traitement d'image sous Android en 3ème année de licence. Cette version est la version complétée du cahier. Le projet est réalisé en binôme.

Description

Le but de ce projet est de développer une application de traitement d'image sur smartphone avec système Android. Les images peuvent aussi bien être obtenues depuis la galerie du téléphone que directement depuis la caméra. Le projet est découpé en deux parties, et deux releases du logiciel seront donc à rendre. La première release est commune à tous les groupes. Elle est composée des fonctionnalités de base afin de gérer, afficher et sauvegarder les images. De plus seront intégrés quelques traitements d'image basés sur une transformation d'histogramme ou une convolution. Avec cette première release, chaque groupe rendra un cahier des charges complété, avec un ensemble de fonctionnalités additionnelles choisies parmi une liste de choix.

Besoins fonctionnels

Les besoins fonctionnels sont donc découpés en deux parties : ce cahier des charges contient les besoins fonctionnels de la première release.

Charger une image

Le logiciel permet d'obtenir une image de plusieurs manières :

- Depuis la galerie : Le logiciel permet de sélectionner une image parmi celles présentes dans la galerie du téléphone.
- Depuis la caméra : Le logiciel permet de capturer une image depuis la ou les caméra(s) du téléphone.

Afficher une image

Une fois une image chargée, le logiciel l'affiche sur l'écran du téléphone.

Zoomer

Lorsqu'une image est affichée dans le logiciel, il est possible de zoomer et dézoomer, en utilisant l'interaction avec deux doigts.

Scroller

Lorsqu'une image est affichée et déborde de l'écran, il est possible de déplacer la zone affichée à l'aide d'une interaction avec un doigt.

Appliquer des filtres

Le logiciel permet d'appliquer quelques filtrages usuels sur les images chargées.

Régler la luminosité

Description

Lorsqu'une image est affichée il est possible d'en régler la luminosité grâce à une barre de progrès.

Fonctionnement

L'interface utilisateur possède une barre permettant de régler le pourcentage de luminosité à réduire. La barre allant de 0 à 100, il suffit de mettre par exemple 50, pour réduire de 50% la luminosité de l'image. Le programme va en réalité calculer 50% de chaque pic de couleur (rouge, vert et bleu).

Égaliser l'histogramme

Description

Le logiciel peut égaliser l'histogramme d'une image en niveaux de gris afin de la rendre plus agréable à l'œil (si trop sombre ou trop lumineuse).

Fonctionnement

Tout d'abord, il est important de calculer l'histogramme de l'image pour l'égaliser. Mais il faut également l'histogramme cumulé, calculé à partir de l'histogramme de l'image. Ensuite, il faut effectuer calculer l'égalisation comme ceci :

$$I'(x, y) = C(I(x,y)) * 255 / N$$

(Où I' est le nouveau pixel à la position (x, y), I le pixel actuel à la position (x, y), C l'histogramme cumulé et N le nombres de pixels de l'image).

Filtrage couleur

Description

Le logiciel permet de mettre en œuvre les traitements de couleur vus en TP, à savoir modification de la teinte (niveaux de gris, sépia, ...) ainsi que la sélection d'une teinte à conserver lors du passage en niveaux de gris.

Fonctionnement

Chaque filtre possède différents algorithmes :

Transformation d'une image en niveaux de gris : Cette méthode va égaliser les niveaux de rouge (R), de vert (G), et de bleu (B) en suivant ce calcul :

R = G = B = 0.299 * R + 0.587 * G + 0.114 * B

Ensuite, en utilisant la méthode des Look-Up Tables, le programme va étirer les niveaux de gris de manière à obtenir des niveaux de gris allant de 0 à 255 systématiquement.

Transformation d'une image en sépia : Cette méthode fonctionne de manière similaire à la méthode des niveaux de gris, à l'exception du calcul des Look-Up Tables qui n'est pas utilisé et, afin d'obtenir les couleurs du sépia, il faut ajouter aux couleurs rouge et vert d'une certaine profondeur (définie à 20 dans notre programme) afin d'obtenir une couleur « jaune pâle » sur l'image.

Isolation d'une couleur pour ne garder qu'elle : Cet algorithme va trouver, en fonction d'une couleur donnée (rouge, vert ou bleu), les pixels où la quantité de couleur est plus importante que les deux autres. Dans ce cas là il ne se passe rien. Sinon, le nouveau pixel devra être sous la forme d'un niveau de gris. Ainsi, l'image n'aura qu'une seule couleur principale restante, qui sera la couleur choisie par l'utilisateur.

Convolution

Description

Le logiciel permet d'appliquer différents filtres basés sur une convolution sur l'image affichée. Les filtres moyenneur, Gaussien, Sobel et Laplacien peuvent être choisis par l'utilisateur.

Fonctionnement

Pour que l'algorithme fonctionne, il est important de choisir un masque de convolution. Un masque de convolution est une matrice d'entiers, carrée et de taille impaire. Ensuite l'algorithme va traverser chaque pixel un à un à l'exception des bordures, et, pour chacun, va appliquer la convolution en utilisant les pixels voisins du pixel sélectionné dans un rayon de la moitié de la taille du masque. Ensuite, le nouveau pixel sera calculé en fonction de la valeur du masque pour former une nouvelle image.

Les bordures ne sont pas retouchées lors de l'application de la convolution.

Effet de dessin au crayon

Description

Le logiciel dessine l'image à la manière d'un croquis au crayon de papier.

Fonctionnement

L'algorithme suit une méthode expliquée par Jin Zhou and Baoxin Li en suivant cette source.

Il faut tout d'abord passer l'image en niveau de gris. Ensuite, il faut appliquer un filtre de Convolution avec le masque Laplacien afin de dégager les contours. Puis, il faut récupérer les valeurs négatives retournées et les transformer en valeurs blanches (255 partout), enfin, il faut transformer le gradient en faisant le calcul 120-g (où g est le gradient) si g est strictement supérieur à 0 et g = 255 sinon.

Dans notre programme, notre méthode a été adapté à nos problèmes (voir les problèmes rencontrés).

Réinitialiser

Le logiciel permet de réinitialiser l'image, c'est-à-dire d'annuler les effets appliqués depuis le chargement de l'image.

Sauvegarder une image

Le logiciel permet de sauvegarder une image modifiée.

Architecture

Dans le cadre de notre architecture, nous avons décidé de programmer une activité principale, contenant un viewer d'image et les boutons permettant d'appliquer les filtres.

Nous avons ajouté, comme demandé une interface contenant les filtres ainsi qu'une classe Image implémentant les filtres.

Une image hérite également d'un bitmap, classe native de java, qui représente une image. On pourra ainsi utiliser la fonction setImageBitmap(Bitmap b) de la classe ImageView, qui permet d'afficher une image sur l'application. Le fait d'utiliser une classe Image permet d'éviter de faire les calculs dans l'activité principale et donc de rendre le code de cette classe trop rempli et peu lisible.

Nous n'utilisons pas les renderscripts malgré leur utilité dans la vitesse d'exécution du programme.

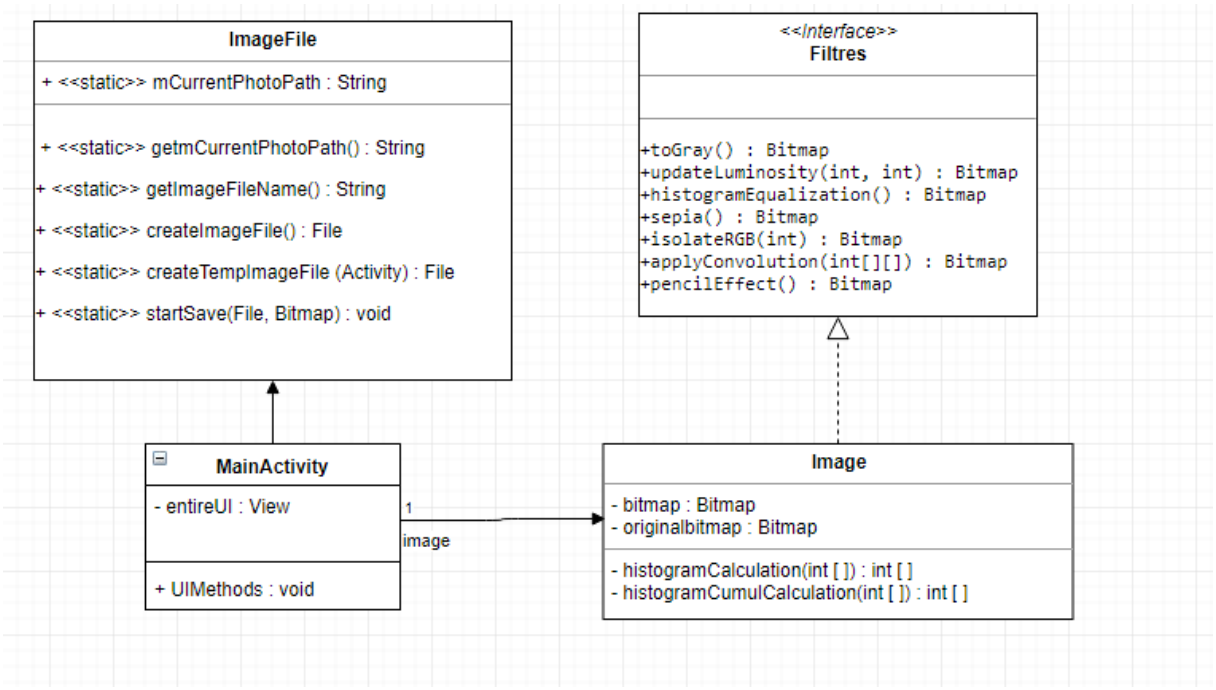

Figure 1 Diagramme de classe de l'application

Tests de compatibilité

Nous avons testé notre programme sur 3 téléphones :

- Un Samsung Galaxy J3 2016 : Version d'Android 5.1 (Lollipop).
- Un Sony Xperia X : Version d'Android 8.0 (Oreo).
- Un Razer Phone : Version d'Android 7.1 (Nougat).

A notre connaissance, aucune de ces 3 versions ne présentaient de problèmes d'exécution.

Difficultés rencontrées et problèmes

Au niveau du code

Les renderscripts

Durant ce projet, notre plus grosse difficulté a été de manipuler les renderscripts. Nous ne comprenions pas réellement comment ils fonctionnaient et quand il aurait fallu s'y pencher afin de les implémenter, nous rencontrions différents problèmes sur notre programme. Ainsi, lorsque l'on a commencé à travailler dessus, il était trop tard pour pouvoir créer un travail correct en les utilisant.

Les erreurs algorithmiques

Dans cette partie, nous allons évoquer différentes erreurs, qui sont pour la plupart résolues dans la version finale.

- L'égalisation d'histogramme : pendant un temps, l'égalisation d'histogramme, qui utilise des couleurs en RGB (sans valeurs négatives) retournaient des valeurs négatives qui n'étaient pas censées exister. Cela était dû à une erreur de calcul lors de l'écriture de l'algorithme dans le calcul du nouveau pixel.
- Le bouton de sauvegarde : lors de sa première implémentation, le bouton de sauvegarde d'une image sauvegardait uniquement la photo originale et non la photo modifiée. Désormais, la photo est correctement sauvegardée dans le téléphone.
- L'effet de crayon : Cette méthode est la seule possédant un problème que nous n'avons pas réussi à résoudre, à savoir un problème lors de l'implémentation de la méthode suivie puisque le filtre laplacien ne faisait apparaitre aucune valeur négative, or il affichait des pixels corrompus. Nous avons donc décidé de modifier notre implémentation, ce qui donne désormais un effet plus proche d'un dessin au burin qu'au crayon de papier.

Au niveau de l'organisation

Étant donné que nous ne nous connaissions pas comparé à la plupart des groupes de projet, il était assez dur pour nous de s'organiser convenablement. Même si nous sommes restés respectueux vis-à-vis de l'autre, il était difficile de créer une relation de confiance mutuelle mais aussi de s'organiser dans la liste des tâches à faire, ce qui nous a fait perdre un temps considérable.

Capitalisation d'expérience et conclusion

En conclusion, ce projet a été très instructif pour nous, puisqu'il nous a initié à travailler avec des personnes que l'on ne connait pas, ce qui est rare dans le domaine scolaire/universitaire. De plus, ce projet nous a appris à gérer notre temps grâce au deux rendus différents, nous poussant à ne pas attendre les derniers jours pour boucler le projet (même si on l'a fait pour chaque rendu).

Le projet en lui-même nous a fait appliquer les différents algorithmes vus en cours, et nous a également appris à rechercher et appliquer des algorithmes tel que celui de dessin au crayon.

Le projet était donc intéressant, demandant en temps en réflexion, et donc un projet donnant une première approche de l'analyse d'image et de la programmation mobile.