Traitement d'image sous Android
Cahier des charges

Thomas Fleury, Gauthier Lamarque, Maxime Mayolini

1 Introduction

Ce document contient une description du travail à rendre dans le cadre du projet de traitement d'image sous Android en 3ème année de licence.

2 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.

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.

Nous utilisons un git composé de deux branches : une contenant l'application final qui nous permettra de créer la release du projet (master) et un qui nous permet de faire des tests sur le traitement d'image.

3 Besoins

Les besoins sont découpés en deux parties : une liste de besoins complétés (c'est-à-dire implémentés et utilisables), et une liste de besoins encore en phase de développement.

3.1 Complétés

1. **Charger une image** Le logiciel permet d'obtenir une image de plusieurs manières :
   (a) **depuis la galerie** Le logiciel permet de sélectionner une image parmi celles présentes dans la galerie du téléphone
   (b) **depuis la caméra** Le logiciel permet de capturer une image depuis la ou les caméra(s) du téléphone
2. **Afficher une image** Une fois une image chargée, le logiciel affiche sur l'écran du terminal
3. **Zoomer** Lorsqu'une image est affichée dans le logiciel, il est possible de zoomer et dézoomer, en utilisant l'interaction avec deux doigts.
4. **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.
5. **Appliquer des filtres** Le logiciel permet d'appliquer quelques filtrages usuels sur les images chargées.
   (a) **Régler la luminosité** Lorsqu'une image est affichée il est possible d'en régler la luminosité
   (b) **Régler le contraste** Lorsqu'une image est affichée il est possible d'en régler le contraste

(c) **Égalisation d'histogramme** Le logiciel permet d'égaliser l'histogramme de l'image affichée

(d) **Filtrage couleur** Le logiciel permet de mettre en oeuvre les traitements 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.

(e) **Convolution** Le logiciel permet d'appliquer différents filtres basés sur une convolution sur l'image affichée. Les filtres moyenneur et Laplacien sont implémentés.

6. **Réinitialiser** Le logiciel permet de réinitialiser l'image c'est-à-dire d'annuler les effets appliqués depuis le chargement de l'image. L'annulation se fait par "couche", c'est-à-dire que l'annulation annule le dernier filtre appliqué sur l'image.
7. **Sauvegarder une image** Le logiciel permet de sauvegarder une image modifiée.

3.2 En cours de développement

8. **Filtres en cours** Certains filtres sont encore manquants ou non-fonctionnels. On pourra noter le filtre Gausien ainsi que le Sobel.
9. **Simuler un effet dessin au crayon** Le logiciel permettrera à partir d'une photo de générer un dessin au crayon représentant l'image initiale.
10. **Restreindre la zone d'application d'un filtre avec le doigt** Le logiciel permettrera de restreindre l'application d'un filtre uniquement autour d'une zone définie par l'utilisateur avec son doigt.

4 Architecture

L'architecture de notre projet est découpée en deux parties : une partie traitant l'interface homme-machine ainsi que le design de l'application, et une autre expliquant plus en détails nos choix concernant l'agencement du code.

4.1 Graphique

Pour l'aspect graphique, l'application est composée de deux activités :

— **MainActivity** : cette activité permet, via des boutons, soit de choisir une image depuis la galerie, soit de prendre une photo, soit de récupérer la dernière photo enregistrée depuis l'application ;

— **ImageActivity** : cette activité nous permet de faire tout le traitement d'image, c'est-à-dire afficher l'image en taille réelle, l'agrandir et la parcourir, et enfin appliquer un filtre. Il est possible aussi d'annuler un traitement. Pour cela, nous avons mis en place un vecteur regroupant toutes les anciennes images, ce qui permet d'annuler des traitements jusqu'à l'image d'origine. Cependant, cela pose un problème : le fait de stocker plusieurs images de taille conséquente peut à terme, faire fermer l'application car il n'y a plus de mémoire disponible. La documentaion d'Android propose pour cela d'utiliser la bibliothèque Glide, qui gère justement un stockage intelligent des images lourdes en mémoire.
Une autre fonctionnalité recquise était celle de pouvoir sauvegarder une image sur le téléphone, et donc il suffit d'appuyer sur l'option "Sauvegarder une image" dans le menu des options. Il y a un certain délai avant que l'image ne soit détectée par l'application Galerie, mais elle apparaît instantanément lorsque l'on parcourt les fichiers du téléphone.

4.2 Code

Les fonctions utilisées dans le code sont implémentées dans deux classes :

— **Filter** : regroupe toutes les fonctions correspondant aux différents filtres. Nous avons trouvé plus pertinent de les regrouper en une seule classe plutôt que de faire une classe par fonction, car il y en a un nombre important et chaque fonction est facilement retrouvable dans le classe ;
— **CustomImageView** : il s'agit de la classe personnalisée qui nous sert à afficher une image dans sa taille réelle. De ce fait, il est possible que nous ne voyons pas l'intégralité de l'image. Du coup, nous avons ajouté ou surchargé des méthodes nous permettant de nous déplacer dans les limites de cette image en taille réelle comme il était demandé, mais aussi de pouvoir zoomer afin d'observer certains détails possiblement mis en valeur par des traitements. Contrairement à ce qui a été vu en cours, Android (ou Java) ne nous laisse pas toucher à la théorie du zoom, ce qui rend son implémentation extrêmement simple. En effet, l'objet GestureDetector va s'occuper de la détection du "pinch", c'est à dire le mouvement des deux doigts associé au zoom, et il ne nous reste plus qu'à définir le nouveau facteur de zoom et ses limites (définies dans notre code en 1 et 5). La méthode "onDraw()" qui s'occupe de redesssiner l'image après un changement va tout simplement appliquer le nouveau facteur de zoom.
L'implémentation du déplacement dans l'image est une autre paire de manches. Dans ce cas là, ce sont les calculs des limites, mais aussi collecter les informations nécessaires pour trouver ces limites qui a été assez laborieux. En effet, on peut voir dans la méthode "onDraw()" citée plus haut, que l'on récupère des informations sur la taille de notre ImageView personnalisée. C'est dû au fait que l'on ne peut pas récupérer la taille de cet élément avant qu'il n'est été "dessiné" à l'écran. De ce fait, collecter les dimensions dans la méthode de dessin était nécessaire.
Il nous reste cependant à traiter les déplacements d'une image zoomée, à l'heure actuelle, le déplacement ne fonctionne que lorsque l'image n'est pas zoomée. Et le cas d'une image plus petite que la vue personnalisée n'est pas traité non plus.

N ous avons aussi d'autres classes qui sont des classes servant uniquement pour l'interface Homme-Machine. Certaines permettent à l'utilisateur de faire un choix (comme MatrixChoice_DialogFragment ou Filter_DialogFragment) et d'autres permettent de choisir sur une barre de recherche (comme SeekBarDialog ou SeekBarColorDialog).

5 Calendrier

Le projet démarre le 20 janvier 2017 et dure 3 mois.

5.1 1er rendu - 3 Mars

Le premier rendu comporte une première release du logiciel, qui comprendra tous les besoins complétés.

5.2 Rendu final - 14 Avril

Le rendu final contiendra le logiciel terminé avec toutes les fonctionnalités, ainsi qu'une série de tests garantissant le bon fonctionnement du logiciel, et une évaluation de ses performances en temps et en mémoire