Rapport de projet
Retouche d'images sur Android

Manuel Ricardo Guevara Garban
Loïc Lachiver
Romain Pigret-Cadou
Sofiane Menadjlia

L3 Informatique

Projet technologique

2020

Table des matières

1 Introduction :

Ce rapport synthétise notre travail de développement d'une application de retouche d'image exécutable sur la plateforme mobile Android.
Réalisé dans le cadre de l'UE *Projet technologique* lors du 2ème semestre de Licence 3 informatique à l'Université de Bordeaux, les objectifs de ce rendu de groupe étaient les suivant :

- Réaliser une application graphique Android
- Permettre de charger, éditer et sauvegarder facilement une image
- Proposer des effets par simple modification de pixels
- Proposer des effets manipulant des histogrammes
- Proposer des effets de convolution
- Utiliser la technologie d'accélération RenderScript
- Gérer le développement d'un projet de groupe

➔Lien GitHub du projet : https://github.com/picachoc/pimp-android

2 L’application :

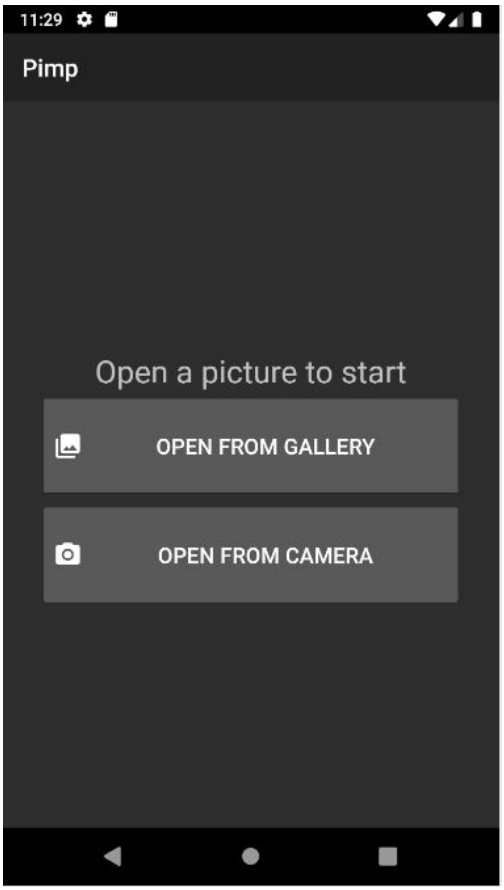
Version minimale Android requise : Oreo (8.0) (API level 26)

2.1 Lancement :

Le projet n’a pas encore été conçu pour être disponible sous un format APK.
Le dépôt git est conçu pour être importé dans Android Studio.
Après avoir cloné le dépôt, rendez vous sur Android Studio puis **File > New > Import Project** et sélectionnez le dossier créé par le clone.

2.2 Utilisation :

A l’ouverture de l’application vous obtiendrez l’interface suivante :



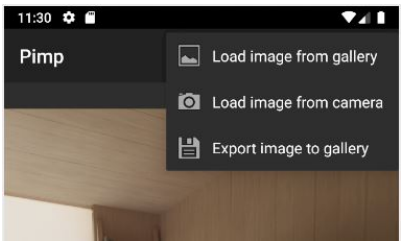
Appuyez sur le premier bouton (🖼) pour ouvrir votre galerie photo et choisir une image à importer.

Ou appuyez sur le second bouton (📷) pour ouvrir la caméra de votre appareil Android. Capturez alors une image, elle sera sauvegardée dans votre galerie et instantanément importée dans l’application.

Il sera nécessaire à la toute première utilisation de ces fonctionnalités d’autoriser l’application à accéder à la galerie et/ou la caméra.

Après avoir choisi une image vous accéderez à la page d'édition principale :

Il vous sera possible d'importer une nouvelle image depuis le menu ⏷ en haut à droite :

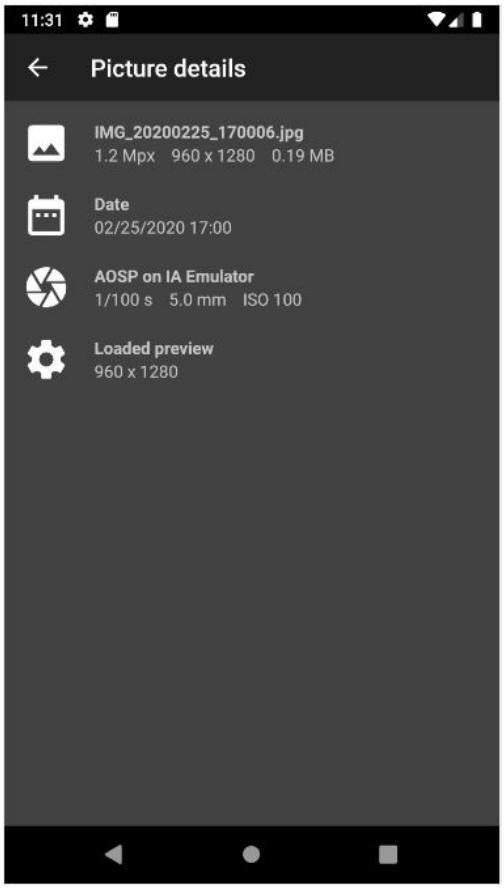


Le bouton 🖫 permettra d'exporter votre image dans votre galerie, cette opération peut prendre un peu de temps car l'image exportée sera de même taille que l'image d'origine.

En bas de l'écran vous trouverez la liste des effets disponibles applicables sur votre image, faites défiler de droite à gauche.

Le bouton ⮌ annule tout les changements appliqués à l'image en la restaurant à son état d'origine.

Le bouton ℹ quand à lui affiche les informations suivantes à propos de l'image :



Les sections 🖼 📅 🌐 et 📍 permettent d'afficher des informations respectivement sur le fichier image, sur sa date de prise de vue, sur l'appareil source et sur ses coordonnées géographiques.
Ces sections ne sont pas toujours visibles et dépendent du fichier image.

Quand à la section ⚙ elle donne les dimensions de l'aperçu actuellement affiché dans l'application, il peut être plus petit que l'image d'origine afin de préserver la mémoire du téléphone et le temps d'exécution. L'image exportée en revanche sera aux bonnes dimensions.

Après avoir sélectionné un effet sur la liste en bas de l'écran, des éléments d'interface vont se superposer à votre image :

Le bouton ✖ en haut à gauche vous fait revenir à la liste d'effets et annule l'effet courant.

Le bouton ✔ en haut à droite applique l'effet et vous fait revenir à la liste d'effets.

En bas des curseurs ou des boutons peuvent apparaître, selon l'effet sélectionné. Manipulez les pour voir en temps réel l'impact sur l'image.

NB : L'entièreté de cette application est verrouillée en mode portrait.

3 Effets :

Interface :

L'utilisateur choisit un effet parmi la liste d'effets affichée en bas de l'écran, ce qui affiche les réglages disponibles (s'il y en a). L'utilisateur peut ensuite choisir de confirmer ou d'annuler la modification apportée par l'effet choisi.

Structure du code :

Les effets sont rassemblés dans le package filters (fr.ubordeaux.pimp.filters). La classe *Retouching* contient les réglages de luminosité, contraste, saturation et teinte. La classe *Convolution* contient tous les effets liés à la convolution (flou, détection de contour etc.). Toutes ces méthodes sont appelées lors de l'appui de boutons ou de glissement de seekbars. Les seekbars ont généralement une étendue allant de 0 à 255 (sauf pour les réglages de teinte), et certains effets nécessitent des valeurs pouvant être négatives. Ainsi la valeur de seekbar est modifiée dans la méthode appelante de ces mêmes effets.

3.1 Luminosité (Brightness) :


Méthode appelante : Retouching.setBrightness()
Script : brightness.rs

Ce réglage se contente d'ajouter une valeur (positive ou négative) aux trois canaux RGB de l'image. Cette valeur est fixée par la seekbar. Les valeurs sont tronquées entre 0 et 255, par conséquent on perd de l'information dans les valeurs extrêmes de luminosité.

Cet effet n'utilise pas la luminosité existante de l'image, ainsi on peut obtenir des résultats qui sont parfois discutables, par exemple le noir qui s'éclaircit et inversement pour le blanc. Pour pallier à ce problème, on pourrait introduire une multiplication afin de modifier la luminosité propor-

tionnellement à celle existante. Cependant, cette solution modifie aussi le contraste, nous avons donc choisi de laisser l'algorithme tel quel.

3.2 Contraste (Contrast) :

Méthode appelante : Retouching.dynamicExtensionRGB()
Script : dynamicExtension.rs

Ce réglage effectue une extension linéaire de dynamique. Les nouveaux extremum de l'histogramme sont définis à partir de la position de la seekbar. La dynamique est ainsi étendue autour d'une valeur se situant au milieu des deux anciens extremum de l'histogramme*. On a donc une image uniforme lorsque l'on règle le contraste au minimum.
Le principal problème de l'extension dynamique est qu'elle ne permet pas d'augmenter le contraste à des grandes valeurs. En effet, les extremum sont vite atteints. La solution serait de faire une transformation linéaire par morceaux.

*Il serait peut-être plus juste de prendre la médiane de l'histogramme cumulé afin d'avoir une valeur qui représente mieux la "valeur moyenne" de l'image.

3.3 Saturation (Saturation) :

4 Structure du projet :

4.1 Structure graphique Android et navigation :

Pour afficher certains éléments d'interface comme par exemple les informations de l'image (bouton ℹ) nous utilisons des **Fragment**. En effet une activité supplémentaire n'est pas nécessaire car cette petite partie de l'application ne correspond pas à un point d'entrée de l'application. Par ailleurs changer de fragment (plutôt que de changer directement de layout) pourrait faciliter l'implémentation d'une interface différente, pour tablette par exemple.
On notera que dans la structure actuelle de l'application, l'image actuellement éditée est contenue et manipulée depuis l'activité principale. Les fragments n'apportent à l'application que des éléments d'interface.

4.2 Classe Image :

Cette classe a été conçu comme une alternative à l'utilisation directe de la classe **Bitmap** fournie par Android.
Le coeur de la classe est évidement une instance de Bitmap, qu'il est possible de récupérer à tout moment. Par ailleurs la classe offre des fonctionnalités supplémentaires, parmi celle ci notamment la possibilité de restaurer l'image à son état au moment de sa création ou de son chargement, via la méthode **reset()**.
On notera la nécessité pour Image d'avoir la référence d'une Activité de l'application, en effet elle est requise à plusieurs moments par les librairies Android pour charger la Bitmap en mémoire.

4.2.1 Classe ImageInfo :

La classe Image 4.2 génère et garde une instance de la classe **ImageInfo**, cette classe contient un grand nombre de valeurs à propos de l'Image (dimensions, coordonnées GPS, date de prise de vue, ...).
L'idée de cette classe est d'empaqueter toutes ces informations afin de faciliter le passage de ces informations à l'avenir, notamment à travers des Fragments ou des Activités. On notera que tous les accesseurs appliquent des opérations de formatage sur ces données, certaines opérations pourraient être déplacées dans les constructeurs s'ils venaient à être utilisés régulièrement.

4.3 Package util :

Ce package contient de nombreuses classes contenant des méthodes statiques permettant une meilleur factorisation du code.
La classe **Utils** offre par exemple des méthodes pour récupérer la taille de l'écran ou pour calculer un ratio de redimensionnement.
La classe **BitmapIO** permet d'effectuer le chargement d'une Bitmap de plusieurs manières, depuis les resources ou un autre emplacement du téléphone, et avec la taille voulue.

5 Performances :

5.1 Temps d'exécution :

TODO

5.2 Mémoire :

TODO montrer l'ancienne instance d'image se faire garbage collecter ? voir remarques

TODO

6 Remarques et améliorations :

6.1 Remarques sur le code

TODO Remarques poids mémoire de la copie orginale des Images.

Lors du chargement d'une nouvelle image, nous ré-instancions un objet de la classe Image. Ce qui veut techniquement dire que jusqu'au prochain passage du ramasse miette Android, deux images sont en mémoires, donc deux Bitmap et deux tableaux de pixels (la copie originale des Images, voir 4.2). C'est un élément discutable cependant notre application limite la taille des Images chargées. Ce qui évite largement les dépassements mémoire.

6.2 Remarques sur les librairies Android

TODO Remarques sur l'utilisation obligatoire d'une activité contexte pour charger une Bitmap.

6.3 Améliorations à court terme :

TODO

7 Gestion du projet :

7.1 Organisation générale :

TODO

7.2 Avis personnels :

TODO

8 Conclusion :

TODO

9 Annexes :

Cahier des charges :
https://dept-info.labri.fr/~vialard/ANDROID/cahierDesCharges.pdf

Représentation de la couleur :
Système de couleur RGB :
https://fr.wikipedia.org/wiki/Rouge_vert_bleu
Système de couleur HSV ou HSB :
https://fr.wikipedia.org/wiki/Teinte_Saturation_Valeur

Documentation Android :
Activity :
https://developer.android.com/reference/android/app/Activity
Bitmap :
https://developer.android.com/reference/android/graphics/Bitmap https://developer.android.com/topic/performance/graphics
Gestion des dimensions des Bitmap :
https://developer.android.com/topic/performance/graphics/load-bitmap
RenderScript :
https://developer.android.com/guide/topics/renderscript/compute
Utilisations des API :
https://developer.android.com/about/dashboards

Divers :
Fichiers images et tags EXIF :
https://fr.wikipedia.org/wiki/Exchangeable_image_file_format