

# Rapport de Projet

Samy ABOU AL TOUT

18 mai 2022



## Table des matières

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>État de l'art</b>	<b>4</b>
<b>3</b>	<b>Planning</b>	<b>5</b>
<b>4</b>	<b>Chronologie du projet</b>	<b>6</b>
4.1	Le debut du projet . . . . .	6
4.2	Le codage du projet . . . . .	7
4.3	La finalisation du projet . . . . .	9
<b>5</b>	<b>Mes joies et peines</b>	<b>11</b>
5.1	Mes joies . . . . .	11
5.2	Mes peines . . . . .	12
<b>6</b>	<b>Conclusion</b>	<b>13</b>
<b>7</b>	<b>Annexes</b>	<b>14</b>

## 1 Introduction

Le sujet de ce projet porte sur le traitement d'image et en plus particulier sur les filtres à l'aide de matrices (kernel).

J'ai choisi ce sujet en particulier car c'est un sujet qui est en rapport avec notre quotidien. De nos jours, beaucoup de personnes aiment poster des photos sur instagram, facebook ou tout autre réseau social. Souvent les gens essayent d'avoir des filtres originaux sauf que cela n'est pas toujours possible vu qu'on est assez limité au niveau des filtres en général.

L'avantage de ce sujet est qu'on y retrouve de l'algorithmie comme par exemple la convolution pour pouvoir créer de superbes filtres.

Le but de ce projet est de pouvoir permettre à l'utilisateur de créer ses propres filtres tout en ayant un choix de quelques filtres prédéfinies. Cela permettrait aux gens de pouvoir faire des filtres plus originaux pour leurs photos qu'ils posteront sur les réseaux sociaux.

Le grand challenge de ce projet est de n'utiliser aucune librairie pour la partie traitement d'image. En effet, en général, des librairies comme SDL2 sont utilisées. Pour ce projet , je n'utiliserais que le C pour la partie traitement d'images.

Pour réussir à faire cela, j'utiliserais uniquement le format Bitmap qui permettra de lire toutes les données de l'image et de pouvoir appliquer des filtres sur les images.

Le but de ce projet serait de donner la possibilité d'utiliser cette technique à toutes les personnes d'une manière simplifiée et user friendly.

La beauté du projet est que l'on fait de l'algorithmie tout en obtenant ce côté visuel magnifique. De plus, ce serait un outil très utile pour les créateurs de contenus, voir même pour les gens qui voudraient se spécialiser dans la création des filtres et vouloir revendre leurs filtres qu'ils ont "designés".

## 2 État de l'art

Le traitement des images est l'utilisation d'un ordinateur numérique pour traiter des images numériques à l'aide d'un algorithme. En tant que sous-catégorie ou domaine du traitement numérique du signal, le traitement numérique des images présente de nombreux avantages par rapport au traitement analogique des images. Il permet d'appliquer une gamme beaucoup plus large d'algorithmes aux images que l'on souhaite traiter et peut éviter des problèmes tels que l'accumulation de bruit et de distorsion pendant le traitement. La création et le développement du traitement numérique des images sont principalement affectés par trois facteurs : premièrement, le développement des ordinateurs ; deuxièmement, le développement des mathématiques (en particulier la création et l'amélioration de la théorie des mathématiques discrètes) ; troisièmement, la demande d'un large éventail d'applications dans l'environnement, l'agriculture, l'armée, l'industrie et la science médicale a augmenté.

Pour l'instant, d'après mes recherches, ce genre de logiciel n'existe pas vraiment. Ceci est plus souvent utilisé par les codeurs, les gens qui ont des connaissances en informatique, mais jamais par des personnes du quotidien qui n'ont pas de connaissances en programmation.

De nombreuses interfaces utilisateur graphiques utilisent des bitmaps dans leurs sous-systèmes graphiques intégrés ; par exemple, le sous-système GDI des plates-formes Microsoft Windows et OS/2, où le format spécifique utilisé est le format de fichier bitmap de Windows et OS/2, généralement nommé avec l'extension de fichier .BMP. Outre le format BMP, d'autres formats de fichiers qui stockent des images bitmap littérales comprennent InterLeaved Bitmap, Portable Bitmap, X Bitmap et Wireless Application Protocol Bitmap. De même, la plupart des autres formats de fichiers d'image, tels que JPEG, TIFF, PNG et GIF, stockent également des images bitmap, mais ils ne sont généralement pas appelés bitmaps, car ils utilisent des formats compressés en interne.

## 3 Planning

### TÂCHES À EFFECTUER

#### 1ERE SOUTENANCE

- N/A
- N/A
- N/A

#### 2EME SOUTENANCE

- IMPLÉMENTER LES FILTRES DE BASES  
(SEPIA, GRayscale, GAUSSIAN...)
- IMPLEMENTER UN ALGORITHME QUI  
PERMET À L'UTILISATEUR DE CRÉER SON  
PROPRE FILTRE

#### 3EME SOUTENANCE

- CRÉER UNE INTERFACE USER FRIENDLY

## 4 Chronologie du projet

### 4.1 Le debut du projet

Pour ce projet, il a fallu faire quelque chose de nouveau ayant un aspect algorithmique et possible à faire en étant seul. Pour cela, à l'aide de Mr. Jimmy Randrianasoa, j'ai décidé de faire un logiciel qui permet d'appliquer des matrices kernel de taille  $3 \times 3$  que l'on désire sur des images BMP et tout cela sans librairie sauf GTK pour l'interface afin de découvrir de nouvelles méthodes et de nouvelles techniques dans le langage C.

Pour pouvoir faire du traitement d'images sans librairies, il a fallu regarder quel format d'images il vaut mieux utiliser afin de pouvoir réussir à appliquer des filtres sur des images. J'ai décidé d'utiliser le format BMP dans lequel on peut assez facilement lire les données de l'image telle que la longueur et la largeur comparé aux autres formats tels que JPEG, PNG qui sont des formats ultra compressés et donc trop complexe à traiter sans librairie.

De plus, j'ai commencé à penser à ce dont j'ai besoin afin de réussir au mieux mon projet de ce semestre. Tout d'abord, un bon IDE comme Xcode afin de travailler et coder dans de bonnes conditions. Puis, j'avais commencé à imaginer l'interface comme par exemple : comment les valeurs de la matrice seront rentrées par l'utilisateur ; si je met à disposition de l'utilisateur de l'application des filtres kernel prédéfinis et si je met des filtres de base tels que grayscale et sepia à disposition. Après cela, je me suis mis à chercher les algorithmes auxquels j'aurais besoin. Je me suis également penché sur la mise en place des calculs de matrices si je privilégie les matrices sous formes de listes chainées ou bien sous formes de tableaux à deux dimensions.

Après avoir bien réfléchi à tout cela et après avoir bien étudié la faisabilité du projet, j'ai élaboré mon plan d'attaque. Je me suis dit que le mieux est de commencer par le traitement d'images puis pour la deuxième soutenance , de faire l'interface la plus user-friendly possible car faire l'inverse m'aurait fait prendre des risques inutiles et m'aurait peut être mis dans l'impossibilité de finir mon projet en temps et en heure et de pouvoir faire quelque chose de correct, de qualité et précis.

## 4.2 Le codage du projet

Pour pouvoir faire un code optimal, efficace et sans librairie, j'ai utilisé des fonctions de base du C.

Par exemple, j'ai utilisé la fonction "getc" afin de pouvoir récupérer les données du header présent dans le fichier Bitmap. La fonction "getc" récupère le caractère suivant (un caractère non signé) du stream spécifié. Cette fonction renvoie le caractère lu sous la forme d'un char non signé converti en un int ou EOF en cas de fin de fichier ou d'erreur.

Pour mettre les données de la nouvelle image dans un nouveau fichier bitmap, j'ai utilisé la fonction putc. La fonction de la bibliothèque C int putc(int char, FILE \*stream) écrit un caractère (un caractère non signé) spécifié par l'argument char dans le stream spécifié. Cette fonction renvoie le caractère écrit sous la forme d'un char non signé converti en un int ou EOF en cas d'erreur.

J'ai utilisé les fonctions "fopen" et "fclose" pour pouvoir ouvrir et fermer les fichiers BMP (les images) sans erreur et en toute sécurité.

Puis il a fallu commencer à voir comment une image BMP est structurée afin d'en extraire les données qui nous intéressent. Les images BMP sont structurées de la manière suivante :

\*En-tête de l'image

\*Table des couleurs (si elle existe)

\*Données de l'image.

L'image BMP comporte un en-tête d'image de 54 octets, une table des couleurs de 1024 octets si elle est présente, et le reste constitue les données de l'image. Après avoir lu les 54 octets, on va extraire de ces 54 octets, les données qui nous intéressent comme la largeur qui se trouve au dix-huitième octet, la longueur qui se trouve au vingt-deuxième octet et le bitDepth qui se trouve au 28ème octet.

Après, il a fallu se pencher sur les filtres de convolutions. Les filtres de convolution (également connus sous le nom de filtre kernel) sont utilisés avec les images pour le flou, la netteté, la détection des bords, etc. Ceci est accompli en effectuant une convolution entre un filtre kernel et une image.

Selon les valeurs des filtres, la convolution peut avoir une variété d'effets. En matière de filtres prédéfinies dans mon application, je donne la possibilité à l'utilisateur de choisir :

\*Blur Kernel

\*Ridge detection( détection des bords et des contours)

J'ai implementé le code qui effectue la convolution après diverses recherches et tentatives de compréhension. J'ai implémenté également deux filtres plus basiques qui n'utilise pas la convolution qui sont :

\*Sepia(qui donne une espèce d'image avec un filtre jaune)

\*Grayscale(qui permet d'enlever les couleurs RGB de l'image)

Pour faire la convolution, on utilise des matrices  $3 \times 3$  de filtre kernel, pour chaque bloc  $3 \times 3$  de pixels dans une image, on multiplie chaque pixel par la matrice  $3 \times 3$ (filtre kernel), puis on fait la somme. Cette somme devient le nouveau pixel. On effectue cela sur toute l'image ce qui nous donnera le résultat que l'on souhaite. Pour les matrices  $3 \times 3$  de filtre kernel, j'ai utilisé des tableaux à 2 dimensions afin de les implémenter en C.

Pendant un certain temps, j'ai fait des fonctions qui marche uniquement sur des images non RGB comme la transformation en noir et blanc, les images négatives. Même si j'avais réussi à les implémenter, je les ai abandonné par la suite car le réel objectif était de travailler sur des images RGB et d'appliquer des filtres de convolutions sur des images RGB. Donc je n'ai que garder que le filtre sépia et le grayscale en filtre de base qui n'utilise pas la convolution.

### 4.3 La finalisation du projet

Afin de pouvoir finir le projet en beauté et de pouvoir lui donner vie afin qu'un utilisateur lambda puisse l'utiliser, il a fallu implémenter une interface de qualité. Pour cela, j'ai décidé de choisir la librairie gtk qui est une valeur sûre pour tout ce qui est interface et GUI.

Pour créer l'interface, j'avais décidé d'utiliser une gridbox afin de pouvoir bien placer mes GtkWidget tels que les boutons classiques, les toggle boutons et les GtkEntry.

J'ai décidé de faire l'interface en anglais afin que tout le monde puisse utiliser l'application vu que l'anglais est la langue que tout le monde parle et que tout le monde comprend.

J'ai fait un bouton "Quit" qui permet de quitter l'application et ferme toutes les fenêtres de l'application qui sont ouvertes. J'ai également fait deux boutons qui permettent d'accéder au site internet officiel du projet et un deuxième bouton qui permet d'être redirigé vers un service d'assistance en ligne si on a une question sur l'application.

Puis, je propose à l'utilisateur de pouvoir sélectionner le filtre kernel qu'il désire ou bien lui donner la liberté de créer son propre filtre kernel en rentrant les valeurs dans la matrice  $3 \times 3$  proposée. Pour cela, j'ai utilisé des GtkEntry pour que l'utilisateur puisse rentrer les valeurs de son filtre kernel de sa matrice "customisée". Pour les autres choix, j'ai utilisé des toggle buttons afin de savoir quelle option l'utilisateur a sélectionné.

Puis, il faut également que l'utilisateur puisse choisir son image BMP. C'est pour cela qu'il y a également un bouton "Choose File" qui permet de choisir l'image que l'on veut. Je l'ai codé en me basant sur ce qui est fourni en exemple dans la documentation de GTK.

Pour finir, il y a le bouton final "Add the filter/kernel" qui permet d'appliquer le filtre et qui ouvre une nouvelle fenêtre où l'on a l'image sans le filtre d'un côté et de l'autre côté, l'image avec le filtre appliqué et choisi par l'utilisateur.

Comme on le sait, la dernière étape a consisté à mettre le code qui applique les filtres avec l'interface. Pour cela, plusieurs sous fonctions qui permettent de gérer ce que va faire le bouton ont été réalisé.

Cependant, le plus dur a été de pouvoir récupéré les données du GtkEntry et les utiliser dans la fonction pour les kernels personnalisés. Il a fallu utiliser la fonction "atof" en C. En effet, la fonction de la bibliothèque C double atof(const char \*str) convertit l'argument chaîne str en un nombre à virgule flottante (type double). Cette fonction renvoie le nombre à virgule flottante converti sous la forme d'une valeur double. Si aucune conversion valide n'a pu être effectuée, elle renvoie zéro (0.0).

Je tiens à préciser que sur les ordinateurs de l'école, il faut rentrer les valeurs avec des "." pour les virgules alors que sur mon ordinateur personnel qui est un Macbook, il faut rentrer les valeurs avec "," pour les virgules sur les chiffres à virgule dans la matrice personnalisé.

Pour éviter que l'application ne crash ou que des segmentations faults apparaissent, j'ai bien évidemment pensé à gérer les erreurs/oublis que l'utilisateur pourrait faire.

Par exemple, lorsqu'un utilisateur oublie de sélectionner le filtre qu'il désire, lorsqu'il va cliquer sur le bouton "Add the filter/kernel", une fenêtre va s'afficher et va lui indiquer qu'il a oublié de sélectionner un filtre. De plus, si jamais l'utilisateur oublie de choisir une image, lorsqu'il va cliquer sur le bouton sur "Add the filter/kernel", une fenêtre va se lancer en lui indiquant qu'il a oublié de choisir une fichier et va lui proposer un bouton dans cette fenêtre qui permet de choisir un fichier afin que le problème/l'oubli soit résolu.

A la fin du projet, j'ai du retravaillé l'interface en utilisant glade puis GTK car les ordinateurs de l'école n'arrivaient pas à supporter mon code de l'interface codé avec la librairie GTK sans Glade malgré que sur mon Macbook, cela marchait très bien. J'ai essayé au maximum de rendre l'interface user-friendly et sympa à utiliser pour l'utilisateur lambda qui ne s'y connaît pas forcément et voudrait tenter de créer des filtres avec des matrices ou bien appliquer des filtres kernels classique ou des filtres basiques sur des images au format BMP de son choix.

## 5 Mes joies et peines

### 5.1 Mes joies

J'ai réelement et beaucoup apprcié d'avoir pu faire ce projet. Cela fut une agréable expérience. J'ai pu découvrir beaucoup de nouvelles fonctions dans le C que je ne connaissais pas avant. En ayant fait seul le projet, j'ai pu découvrir une autre vision dans la manière de réalisation d'un projet informatique ce qui est une chance que je n'aurais plus après et qui me donne donc plusieurs techniques et méthodes de travail.

J'ai appris également que lorsqu'on travaille seul, on doit résoudre ses problèmes tout seul car personne ne sera là pour nous aider. J'ai été très content d'avoir réussi à faire un programme de filtre de convolution qui moi au début , croyait que c'était impossible, j'ai remarqué qu'avec de la volonté et des bonnes recherches, que cela été largement faisable malgré qu'au début j'étais plutot réticent à vouloir faire ce projet.

J'ai été vraiment satisfait de moi car malgré le fait d'être seul, j'ai réussi à surmonter beaucoup de problème. Tout d'abord j'avais réussi à implémenter en C l'application d'un filtre kernel sur une image BMP avec un algorithme de convolution sur une image RGB ce qui n'était pas gagné d'avance vu que j'avais pris du temps à comprendre le format Bitmap, la différence entre les images RGB et les images classiques et comment fonctionne réelement l'algorithme de convolution.

Mais il faut bien l'avouer, le plus satifsaisant est lorsque l'on arrive à faire une interface de qualité et user-friendly, ce qui est en général assez difficile surtout lorsque l'on est seul. J'ai été très content lorsque mon code n'a pas marché sur les ordinateurs de l'école, que j'ai réussi à directement penser à une alternative à mon problème afin d'y remédier au plus vite. Quand j'ai vu le résultat final de mon projet final après plusieurs affinement et réglages, j'étais, je crois l'homme le plus heureux du monde vu que je n'arrivais pas à croire que j'ai pu réussir à faire une application tout seul par moi même sans aide. J'étais extremement satisfait lorsque j'ai vu les comparaisons des images sans filtre kernel ou filtres basiques et quand les filtres sont appliqués. Le meilleur restera d'avoir pu utiliser uniquement le langage C pour le traitement d'image.

## 5.2 Mes peines

J'ai été triste en même temps d'avoir fait ce projet tout seul car je n'ai pas pu connaître la joie de faire un travail en équipe comme lors du projet S2 ou S3. Cette fameuse joie lorsqu'on voit que tous ensemble, on a réussi à s'entraider et accomplir notre mission.

J'ai également été triste lorsque j'ai du changer de projet en plein milieu du semestre car je me suis retrouvé tout seul lors de ce S4. J'ai eu vraiment du mal à savoir quel projet faire tout seul car c'est un choix assez complexe car l'on doit trouver une idée de projet réalisable par une personne et en même temps un projet qui correspond et répond aux critères imposés.

J'ai été dégouté quand j'ai vu que mon long travail pour l'interface n'a pas fonctionné et que j'ai du reprendre à zéro malgré le long travail fourni alors que cela marchait super bien sur mon ordinateur personnel.

Je me suis senti un peu seul dans ce projet vu qu'il n'y avait aucune ambiance de travail de groupe. Personne pour motiver l'autre, si je ne me motivais pas tout seul, rien n'aurait été possible.

C'est vraiment beaucoup plus dur de faire un projet tout seul car on ne peut compter sur personne, on ne peut uniquement compter sur soi-même. Même si le projet était fort sympathique, il restait quand même assez lourd pour une personne vu que j'ai été le seul à tout gérer que ça soit l'interface ou le traitement d'image.

J'ai eu du regret à la deuxième soutenance de ne pas avoir pas fait une présentation qui était vraiment à la hauteur de mon travail au niveau du code, mais cela m'a permis d'apprendre de mes erreurs et de pouvoir faire une présentation digne de ce nom pour la dernière soutenance à la hauteur du travail fourni au niveau du codage et de l'interface. Cependant, je me dis que c'est toujours après les échecs que l'on obtient le succès et la réussite.

## 6 Conclusion

Pour conclure, cela fut assez riche comme projet car il y avait l'aspect algorithmique avec l'effet de challenge de n'utiliser aucune librairie pour le traitement d'image. On avait d'un côté l'aspect algorithmique et mathématique avec la convolution et les matrices  $3 \times 3$ , puis d'un autre côté l'aspect visuel car l'on manipule des images et c'est assez beau comme résultat.

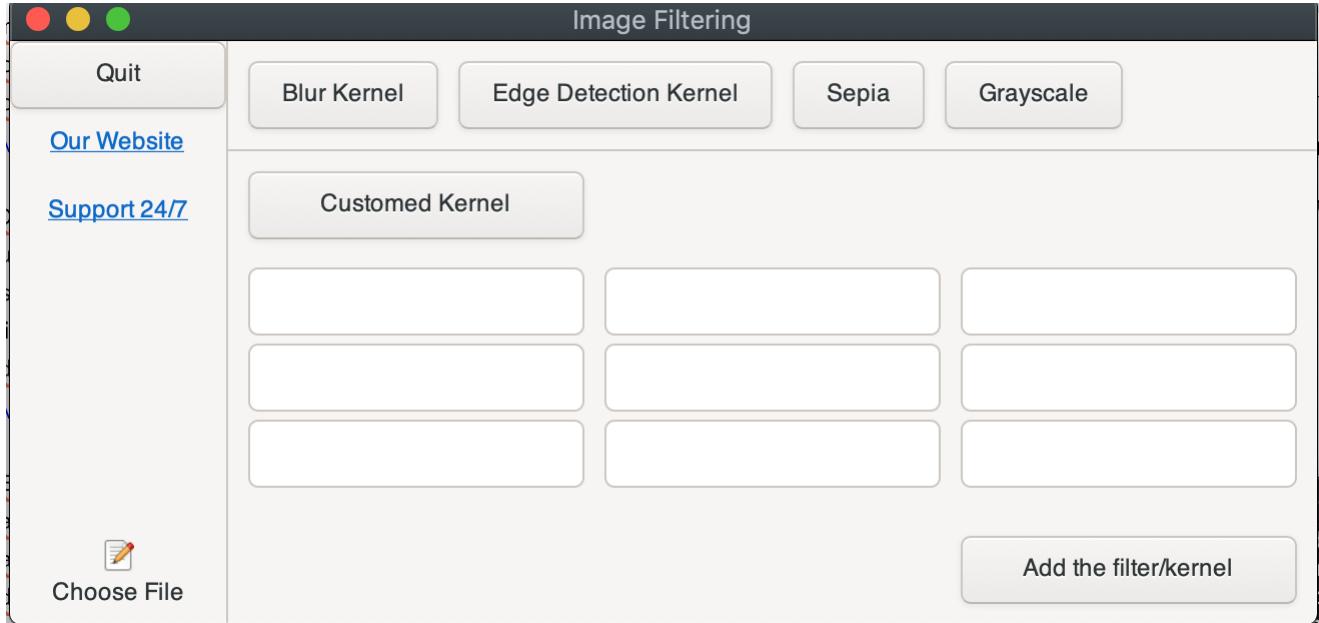
De plus, j'ai été très fier d'avoir mis en place une interface user-friendly et d'avoir réalisé un projet comme celui-ci qui par la suite pourra servir et aider les gens à appliquer des filtres sur des images voir même de créer leurs propres filtres, de s'épanouir et d'utiliser leur esprit créatif.

Et pour finir cette conclusion en beauté, je trouve que j'ai beaucoup appris à travers ce projet, que ce soit au niveau du code en utilisant aucune librairie pour le traitement d'image ou bien, au niveau de la différence entre le travail de groupe et le travail seul sans personne. Je me suis donné moi-même les moyens, la volonté et la motivation pour réussir à réaliser ce magnifique projet.

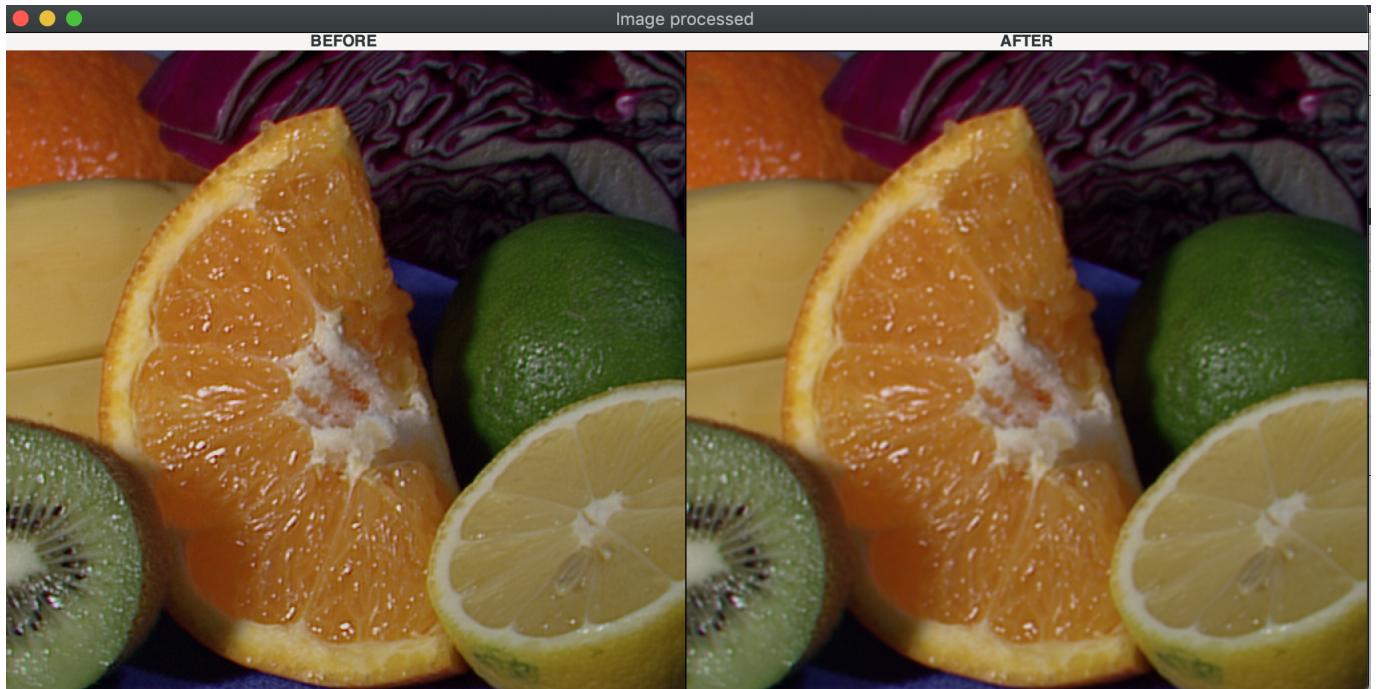
Je tiens à remercier Mr Jimmy Randrianasoa de m'avoir donné l'idée et l'opportunité de faire ce projet.

## 7 Annexes

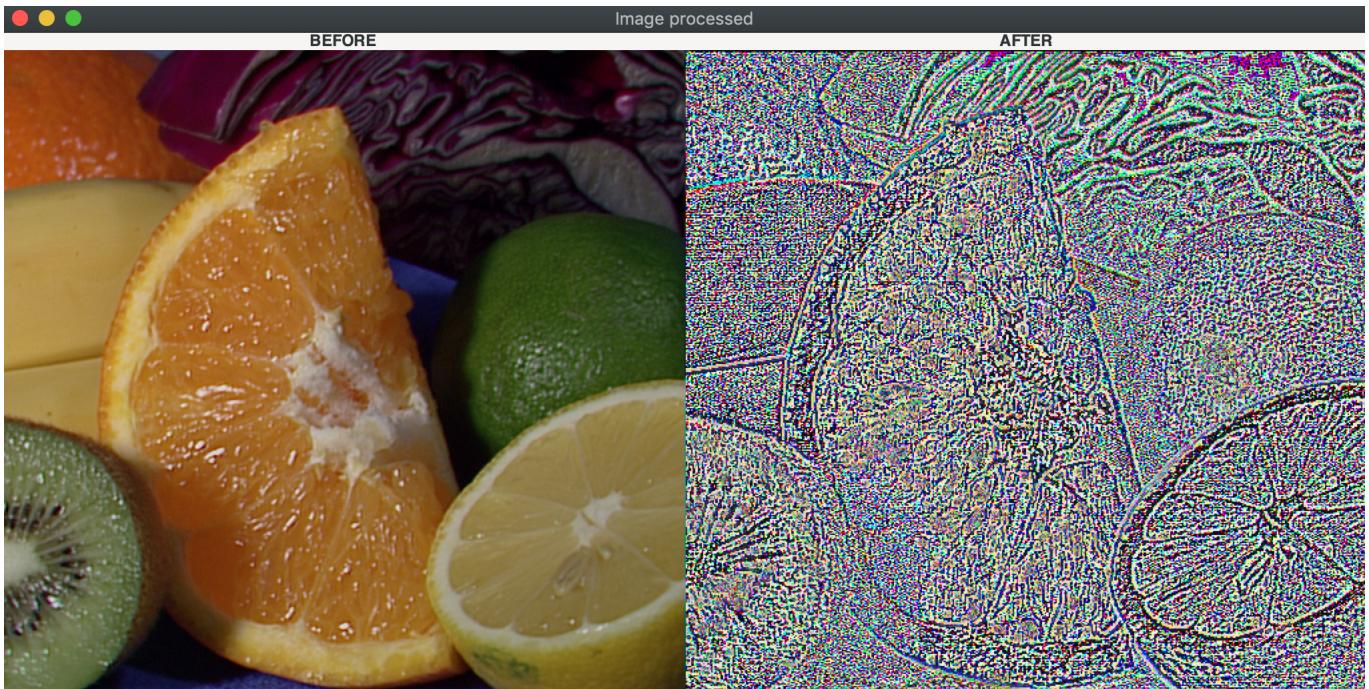
Quelques photos et screenshots de l'application :



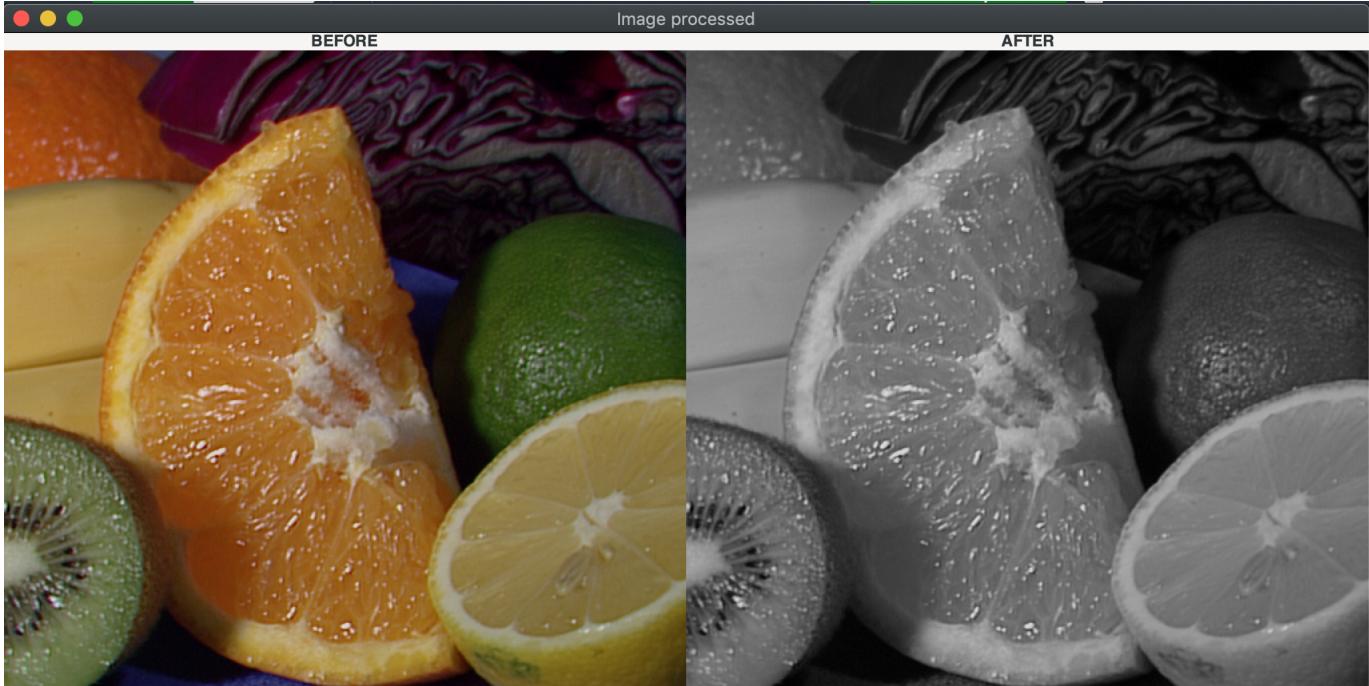
Voilà comment se présente l'interface lorsque l'utilisateur lance l'application.



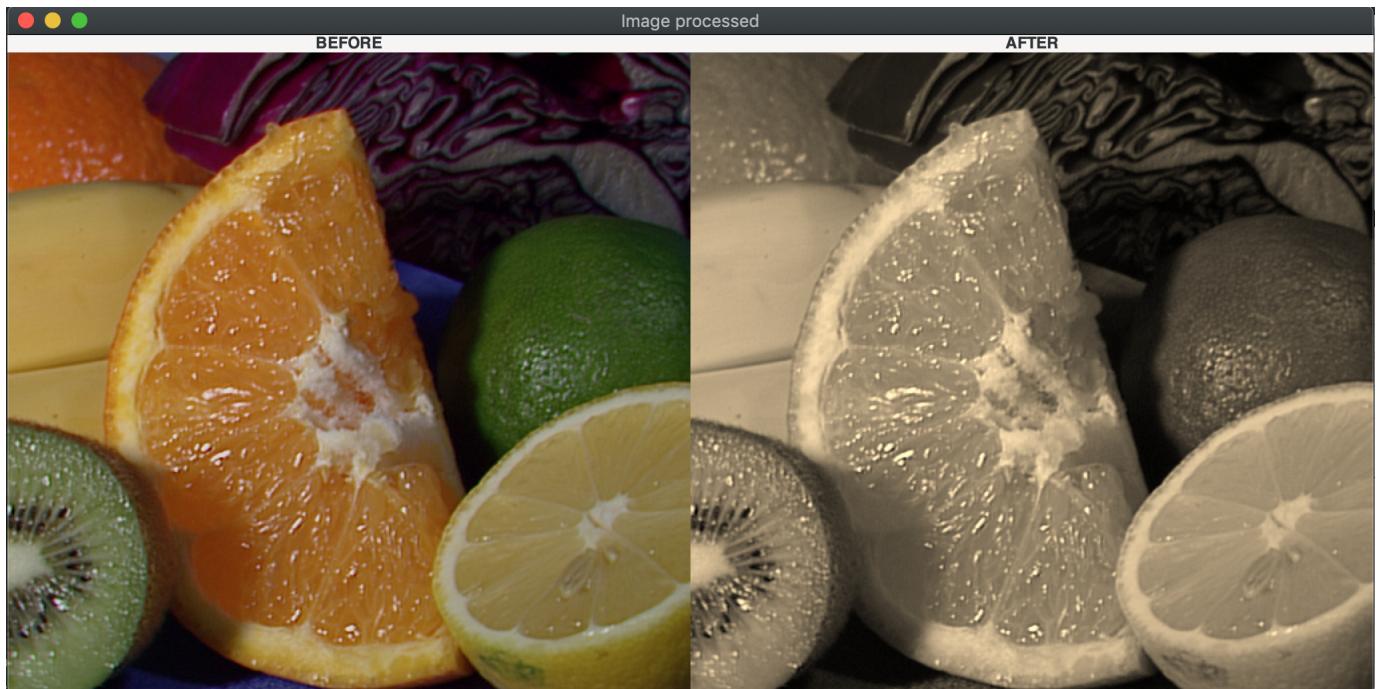
Ici l'application du Blur Kernel.



Ici l'application du kernel qui detecte les bords.



Ici une transformation en grayscale.



Ici une transformation en sepia.