Cahier des charges

Etienne Lazaro, Andrew Youansamouth, Nathan Gillotin

22 janvier 2024

Table des matières

1 Introduction

Lorsqu'on considère les diverses opérations de modification d'images, un éventail de possibilités s'offre à nous, permettant ainsi de façonner et d'adapter les visuels en fonction des besoins spécifiques de l'utilisateur. Parmi ces opérations, la rotation d'image offre la possibilité de donner une perspective nouvelle à une scène, d'explorer des angles inattendus, et ce, sans compromettre la qualité visuelle. Ce processus, bien qu'essentiellement mathématique, a le pouvoir de métamorphoser l'aspect visuel d'une image en jouant avec les positions relatives de ses pixels.

De même, le redimensionnement d'image, un processus de modification de la taille de l'image, se révèle être une démarche incontournable pour ajuster la résolution et l'échelle d'une image selon les exigences spécifiques d'une application. Cette opération, tout en préservant les détails importants de l'image, peut influencer de manière significative la perception visuelle de la scène capturée.

La translation d'image, quant à elle, ouvre la voie à la délocalisation de la scène visuelle dans le plan bidimensionnel. C'est une opération qui permet de déplacer l'image selon un vecteur de translation défini, offrant ainsi une flexibilité accrue dans le positionnement spatial de la représentation visuelle.

Ces opérations, bien qu'elles puissent être mises en œuvre sans l'aide de bibliothèques spécialisées, bénéficient grandement de l'utilisation d'outils tels que Pillow. La bibliothèque Pillow facilite non seulement l'implémentation de ces transformations, mais elle offre également une interface conviviale pour explorer et expérimenter avec divers ajustements visuels.

C'est pour cela que nous, Xuligion, avons décidé de concevoir une application rassemblant toutes ces fonctionnalités. Nous allons proposer à l'utilisateur la possibilité d'éditer ses images à souhait et de les sauvegarder de nouveau sur leur appareil ! Il s'agit donc d'un editeur d'image.

2 Présentation du projet : Editeur d'Images en Rust

2.1 Interface graphique

Afin de pouvoir manipuler et editer les images en toute facilite il est evident qu'il faut une interface graphique. Avec ceci on pourra egalement ajouter d'autres fonctionalites pour accompagner l'utilisateur dans sa demarche. Comme notament :

Une sauvegarde de l'image : Car il faut bien pouvoir enregistrer son travail.

Undo (Ctrl + Z) : Pour revenir en arriere.

Redo (Ctrl + Z) : Pour annuler le retour en arriere.

2.2 Filtrage d'images

Nous allons proposer à l'utilisateur de pouvoir appliquer des filtres sur ses images.

2.2.1 Filtre Médian

Le filtre médian est utilisé pour réduire le bruit dans une image. Il fonctionne en remplaçant chaque pixel par la médiane des valeurs des pixels voisins dans une fenêtre définie.

L'algorithme du filtre médian consiste à trier les valeurs des pixels dans la fenêtre et à prendre la valeur médiane comme nouvelle valeur du pixel central.

2.2.2 Filtre Moyen

Le filtre moyen (ou filtre de lissage) est un filtre qui remplace la valeur de chaque pixel par la moyenne des valeurs des pixels voisins dans une fenêtre définie.

Il est couramment utilisé pour flouter une image et réduire le bruit.

2.2.3 Filtre Gaussien

Le filtre gaussien est un autre filtre de lissage utilisé pour flouter une image. Il utilise une fonction gaussienne pour calculer le poids des pixels voisins.

Les pixels plus éloignés ont un poids moindre, créant ainsi un effet de flou plus naturel que le filtre moyen.

2.2.4 Filtres de Couleur

Les filtres de couleur sont souvent utilisés pour modifier l'apparence chromatique d'une image en ajustant les niveaux de couleur. Par exemple, un filtre rouge peut

accentuer les tons rouges dans une image.

Ces filtres peuvent être appliqués en ajustant les canaux de couleur (rouge, vert, bleu) des pixels de l'image.

2.3 Segmentation d'images

La segmentation d'image consiste à diviser une image en régions ou segments, souvent basées sur des caractéristiques telles que la couleur, la texture, le contraste, etc.

Pour la segmentation, des techniques comme la segmentation par seuillage, la segmentation basée sur la région, la segmentation par contour (comme les contours de Canny), ou même des méthodes plus avancées comme la segmentation par apprentissage profond peuvent être utilisées.

2.3.1 Détection de Région d'Intérêt (ROI)

Une fois l'image segmentée, la détection de la région d'intérêt (ROI) implique de choisir la partie spécifique de l'image que l'utilisateur souhaite analyser.

Les techniques de détection de ROI peuvent être manuelles (l'utilisateur spécifie la région) ou automatiques (utilisation d'algorithmes pour détecter une région d'intérêt en fonction de certaines caractéristiques).

2.4 Transformation d'images

Permettre à l'utilisateur de pouvoir modifier son ilage en la tournant ou en changeant sa taille noue parait essentiel.

2.4.1 Redimensionnement d'image

Le redimensionnement change la taille de l'image en ajustant la distance entre les pixels. Il peut être réalisé en recalculant les valeurs des pixels selon les nouveaux paramètres de taille. En réalité il existe deux façon de redimensionner une image, l'une conserve les donées des pixels a l'échelle, on apelle cela un rognage d'image, le second fais une moyenne de pixels et en réduit leur quantité, cela s'apelle une réduction d'image. nous allons implémenter les deux.

2.4.2 Translation d'image

La translation déplace l'image le long de l'axe x et y. Cela peut être accompli en ajustant les coordonnées des pixels en fonction du vecteur de translation. une translation peux s'apparanter a un rognage d'image, c'est pour cela que les fonction de redisionnement et de translations serons similaires. l'effect de translation trouveras surtout son utilité lorsque nous superposerons plusieurs images

2.4.3 utilité

Ces trois fonctions trouveront leur pleine utilité une fois implémenté dans une interface graphique. comme vu précédament, et surtout pour la translation d'image, certaines fonctions ne peuvent pas donner de résultats interessant, puisque peu utilisable tel quel pour n'importe quel objectif, mais leur utilité deviendront très importante une fois qu'ils peuvent s'utiliser avec aisance, comme lorsque utiliser dans une interface graphique.

2.5 Défis techniques

Nous envisageons aussi certains défis techniques à relever. Parmi eux, celui qui ressort le plus n'est autre que l'optimisation. En traitement d'image, pouvoir en proposer un qui soit aussi rapide qu'efficace est essentiel. C'est pour ça que nous souhaitons mettre un point d'honneur à cela. Malgré notre méconnaissance de Rust, nous souhaitons l'optimiser pour garantir des performances élevées, en utilisant des fonctionnalités avancées comme les traits, les itérations rapides...

De même, afin d'obtenir un outil scientifique efficace, il se doit d'être utilisé par un plus grand nombre dans leur langue maternelle. C'est pour cela que nous avons choisis de fournir des interfaces pour permettre l'utilisation de la bibliothèque avec d'autres langages, en particulier ceux couramment utilisés dans le domaine scientifique.

Enfin, nous souhaitons nous maintenir dans cette optique d'une utilisation facile et accessible à tous en proposant une interface graphique (GUI). Nous ajouterons donc une interface graphique pour permettre aux utilisateurs d'interagir facilement avec la bibliothèque.

Nous accompagnerons donc cela d'une documentation complète et des exemples d'utilisation pour faciliter l'adoption de la bibliothèque par d'autres développeurs.

3 Organisation sur le projet

3.1 Repartition des tâches

	Nathan	Andrew	Etienne
Interface utilisateur			
Interface utilisateur			X
Sauvegarde de l'image	X		
Undo (Ctrl + Z)		X	X
Redo (Ctrl + Shift + Z)		X	X
Filtrage d'images			
Filtre Médian		X	X
Filtre Moyen		X	X
Filtre Gaussien	X		X
Filtre de couleur	X		X
Segmentation d'images			
Segmentation d'image		X	
Detection de Region d'Interet (ROI)	X		X
Transformation d'images			
Rotation d'image	X		
Redimensionnement d'image		X	X
Translation d'image		X	X

3.2 Prévision de l'avancement

3.2.1 Première Soutenance

Interface utilisateur	
Interface utilisateur	0%
Sauvegarde de l'image	100%
Undo (Ctrl + Z)	0%
Redo (Ctrl + Shift + Z)	0%
Filtrage d'images	
Filtre Médian	100%
Filtre Moyen	100%
Filtre Gaussien	100%
Filtre de couleur	100%
Segmentation d'images	
Segmentation d'image	50%
Detection de Region d'Interet (ROI)	0%
Transformation d'images	
Rotation d'image	20%
Redimensionnement d'image	20%
Translation d'image	0%

3.2.2 Deuxième Soutenance

Interface utilisateur	
Interface utilisateur	33%
Sauvegarde de l'image	100%
Undo (Ctrl + Z)	100%
Redo (Ctrl + Shift + Z)	100%
Filtrage d'images	
Filtre Médian	100%
Filtre Moyen	100%
Filtre Gaussien	100%
Filtre de couleur	100%
Segmentation d'images	
Segmentation d'image	100%
Detection de Region d'Interet (ROI)	20%
Transformation d'images	
Rotation d'image	100%
Redimensionnement d'image	100%
Translation d'image	50%

3.2.3 Soutenance Finale

Interface utilisateur	
Interface utilisateur	100%
Sauvegarde de l'image	100%
Undo (Ctrl + Z)	100%
Redo (Ctrl + Shift + Z)	100%
Filtrage d'images	
Filtre Médian	100%
Filtre Moyen	100%
Filtre Gaussien	100%
Filtre de couleur	100%
Segmentation d'images	
Segmentation d'image	100%
Detection de Region d'Interet (ROI)	100%
Transformation d'images	
Rotation d'image	100%
Redimensionnement d'image	100%
Translation d'image	100%

4 Conclusion

Il y a de nombreuses raisons qui nous poussent à choisir ce projet. La première vient tout simplement du traitement d'image en lui-même. Traiter d'images avec du code fait partie des connaissances que nous jugeons essentiel de posséder pour chacun d'entre nous. Avec le projet précédent visant à réaliser une résolution automatique de Sudoku, nous avons eu la possibilité de travailler une première fois dessus sur un projet à grande ampleur. Mais nous estimons que ces connaissances se doivent d'être renforcées.

De même, l'utilisation d'un nouveau langage peut se révéler complexe et fastidieux. Choisir un projet où chacun d'entre nous possède un minimum de connaissance peut nous aider grandement à faciliter notre apprentissage de Rust. En faisant ainsi, nous pourrons alors combiner le renforcement de nos connaissances préalablement existantes en termes de traitement d'image tout en améliorant en tandem notre maîtrise de Rust ainsi que notre capacité à coder avec ce langage.

Enfin, l'application sera utile pour l'utilisateur grace à sa performance, sa facilité d'accès et notament sa prise en main facile à l'aide d'une interface graphique moderne et performante qui lui permettra de modifier et editer des images à souhait dans le cadre de leurs projets. Notre application doit inspirer une prise en main facile et libre d'accès à tous pour en faire un réussite.