Compte Rendu de TP — Reconstruction d’une Fresque

KALALA KABAMBI David

MARTY Julien

Méthodologie

1) Chargement et manipulation des fragments

Chaque fragment est un fichier image ayant un index. Pour obtenir l'image finale, il a fallu :

1. Charger les fragments à partir de fichiers nommés sous le format `frag_eroded_<index>.png`.
2. Lire les informations de placement (`<index> <posx> <posy> <angle>`) depuis `fragments.txt`, ce qui nous indique la position et la rotation exacte du fragment dans l'image finale.
3. Appliquer une rotation à chaque fragment, en fonction de l'angle fourni.
4. Placer les fragments sur une toile vierge ou l'image finale.

Défis rencontrés

Un des principaux défis a été de gérer les pixels noirs présents autour des fragments après leur rotations. En effet, chaque fragment après rotation est entouré d'une large bande de pixels noirs. Les bordures noires des fragments sur les bords de l'image provoquaient des erreurs lorsque l'on plaçait le fragment sur l'image. En effet les fragments étaient bien contenus dans la peinture, mais les bordures noires étaient en dehors de l'image (provoquait des erreurs de type « out of bounds » au moment du placement). Pour cette raison, nous avons décidé de couper l'image au plus près du fragment pour enlever les bordures noires. Ainsi lorsque les fragments sont placés, ils ne débordent plus de l'image.

Cependant, bien que la méthode de recadrage des contours noirs améliore la précision, elle ne permet pas d’éliminer complètement les bordures noires le long des parties courbées des fragments. Le recadrage est efficace pour obtenir une image rectangulaire la plus proche possible du fragment, mais certaines zones noires subsistent, surtout dans les parties arrondies. Les bordures noires restantes, autour des derniers fragments placés sur l'image, recouvraient parfois les fragments placés plutôt. Pour contourner ce problème, un masque a été créé pour éviter de copier les pixels noirs lorsque les fragments sont placés sur la toile.

Il est a noté que les problèmes rencontrés sont facilement évitables avec une implémentation plus

naïve grâce à une double boucle for et en plaçant sur la toile, uniquement les pixels qui ne sont pas noirs. Nous avons cependant fait le choix d'utiliser python qui est un langage qui permet simplement d'utiliser la vectorisation pour réduire grandement les temps de calcul. L'écriture vectorisé nous a ainsi demandée de résoudre les défis mentionnés au-dessus.

2) Reconstruction et visualisation

Une fois les fragments correctement chargés et manipulés, ils sont placés aux positions indiquées dans `fragments.txt` sur une image vierge. La fonction `show_image()` permet ensuite de visualiser l'image finale avec les fragments assemblés.

Évaluation de la qualité de reconstruction

Pour évaluer la précision d'une reconstruction automatique, nous avons comparé les fragments placés dans un fichier `solution.txt` à ceux de la vérité terrain (`fragments.txt`). Un fragment est considéré bien positionné si :

- La différence entre sa position en `x` dans les deux fichiers est inférieure ou égale à un seuil `Δx`.
- La différence entre sa position en `y` est inférieure ou égale à un seuil `Δy`.
- La différence entre les angles est inférieure ou égale à un seuil `Δα`.

Les valeurs par défaut utilisées pour ces seuils étaient `Δx = 1px`, `Δy = 1px` et `Δα = 1°`. La précision est ensuite calculée comme le rapport entre la surface totale des fragments correctement positionnés, moins la surface des fragments mal placés, divisé par la surface totale de tous les fragments présents dans l'image de vérité terrain.

Trois types de fragments :

1. Les fragments incorrects : Des fragments étaient présents dans solution.txt mais pas dans fragments.txt. Ils devaient être considérés comme des erreurs dans la reconstruction.

2. Les fragments mal positionnés : Les fragments présents dans fragments.txt et solution.txt mais mal placé malgré le seuil de tolérance.

3. Les fragments correctement positionnés : Les fragments présents dans fragments.txt et solution.txt correctement placé en prenant en compte la tolérance.

Défis rencontrés

Le principal défi que nous avons rencontré est la récupération de la surface des fragments pour calculer la précision. En étudiant le type d'image à notre disposition, nous avons remarqué que les images étaient entourées non pas de pixels noirs mais transparent. Nous avons alors utilisé cette propriété des pixels pour compter le nombre de pixels composant le fragment. Nous chargeons alors chaque fragment avec le cannal alpha (transparence), puis comptons le nombre de pixels dont la composante alpha est supérieure à zéro (les pixels qui ne sont pas totalement transparents). Ainsi nous obtenons le nombre total de pixels appartenant au fragment.

Après quoi nous comparons la liste des fragments contenu dans solution.txt et fragments.txt. Nous partons du principe que les fragments présents dans solution.txt sont triés dans l'ordre croissant par rapport à leur IDs. A chaque fois qu'un fragment est manquant, mal placé ou bien placé, nous ajoutons le nombre de pixel correspondant au fragment à la liste associée. Enfin nous calculons la précision en utilisant le nombre total de pixel par liste.

Conclusion

Ce TP nous a permis de développer des compétences en manipulation d'images avec OpenCV, en particulier la rotation et le recadrage d'images. Nous avons également pu approfondir notre compréhension des techniques de comparaison et d'évaluation de précision, en mettant en place un système pour évaluer une reconstruction automatique par rapport à une vérité terrain. Bien que certains défis persistent, notamment la gestion des contours noirs après rotation, la solution développée offre une méthode efficace pour l'assemblage et l'évaluation des fragments d'image.

Figure 1 Résultat de la reconstruction de la toile