POIRIER Romain

Rapport de stage : MosaicVision

Département Sciences du Numérique - Deuxième année
2023-2024

Table des matières

1 Introduction

1.1 Mise en contexte

Les mosaïques se composent de tesselles aux formes et tailles variées, souvent irrégulières, et séparées par du mortier dont l'agencement n'est pas nécessairement uniforme. De plus, les couleurs des tesselles sont généralement ternes et présentent un faible contraste, ce qui complique la distinction entre les tesselles et le mortier. La segmentation automatique des tesselles représente donc un véritable défi.

1.2 Objectifs

Ce travail a pour objectif de développer une méthode de segmentation spécifiquement adaptée aux mosaïques, en se focalisant sur l'extraction des tesselles en tant qu'entités distinctes. Une telle approche permettrait aux archéologues d'analyser directement les tesselles d'une mosaïque pour en faciliter la numérisation. Les tesselles détectées serviront de base pour une analyse approfondie, facilitant ainsi l'exploitation des informations extraites.

1.3 Exploration de l'état de l'art

Dans un premier temps, j'ai expérimenté différentes approches classiques de segmentation. Une méthode décrite dans l'article [1] se basait sur la transformation par ligne de partage des eaux pour la segmentation. Cependant, le prétraitement proposé, reposant sur une comparaison des niveaux de gris dans un voisinage restreint, n'a pas permis d'obtenir une image prétraitée satisfaisante. Cela pourrait s'expliquer par la faible distinction de couleur entre les tesselles et le mortier dans nos images de mosaïques. De plus, j'ai testé certaines approches basées sur les superpixels, telles que SLIC et SLIC0, mais les résultats se sont avérés décevants.

J'ai également exploré des méthodes de segmentation classiques appliquées à des objets similaires aux tesselles, comme la segmentation des roches [2]. Toutefois, les techniques de prétraitement basées sur les niveaux de gris n'étaient pas assez performantes, les méthodes telles que Phansalkar ou Otsu n'étaient pas assez spécifiques à notre problème, les résultats n'étaient pas concluants. De plus, les méthodes de détection de contours [3] telles qu'EDlines ou ULSD se sont révélées inefficaces pour ce type d'application.

En somme, les caractéristiques propres aux mosaïques rendent les méthodes traditionnelles de segmentation basées sur les pixels peu adaptées, car elles ne parviennent pas à isoler correctement les tesselles du reste de l'image. Ce travail vise donc à proposer une approche plus robuste, en s'appuyant sur l'intelligence artificielle.

2 Utilisation de Segment Anything

2.1 Introduction

En testant Segment Anything, un modèle d'intelligence artificielle pour la segmentation d'images développé par Meta, j'ai constaté que les résultats obtenus avec cette approche surpassait largement ceux des méthodes classiques. J'ai utilisé la génération de masques fournie par Segment Anything (SAM) pour segmenter mes images. En entrant une image dans le modèle, plusieurs masques (images binaires) des formes détectées sont générés. Lors de l'application de SAM sur l'image complète d'une mosaïque, le modèle détecte non seulement les tesselles, mais aussi des éléments plus grands, comme un personnage présent sur la mosaïque. Cependant, en appliquant SAM sur une partie de l'image dépourvue de formes clairement identifiables, les masques générés étaient nettement plus précis.

2.2 Choix de l'image

Pour obtenir des masques de meilleure qualité, j'ai comparé l'application de SAM sur différentes images. Plus le contraste entre le mortier et les tesselles est marqué, plus SAM est performant pour détecter les tesselles. La segmentation se comporte bien sur l'albédo, mais cette image a tendance à créer des aplats de couleur ou à lisser les textures donc ce n'est pas forcément le meilleur choix pour segmenter une image de mosaique. De plus, j'ai comparé les résultats obtenus avec un fort et un faible éclairage et il est préférable d'avoir un bon éclairage pour obtenir des contours plus précis autour des tesselles. Enfin, j'ai testé la segmentation avec un champ des normales mais les contours n'étaient pas corrects et moins précise.
Les résultats obtenus dépendent fortement de la qualité de l'image originale. Si les tesselles ne se distinguent pas nettement du mortier à l'œil nu, SAM aura probablement des difficultés à les segmenter correctement.

2.3 Tuilage

Puisque la segmentation avec SAM fonctionne bien sur des parties réduites de l'image, j'ai mis en place un tuilage. L'idée est de découper l'image en plusieurs morceaux pour générer les masques sur chacun de ces morceaux, puis d'assembler tous les masques obtenus sur un masque global de la taille de l'image originale, en les positionnant aux bons endroits.

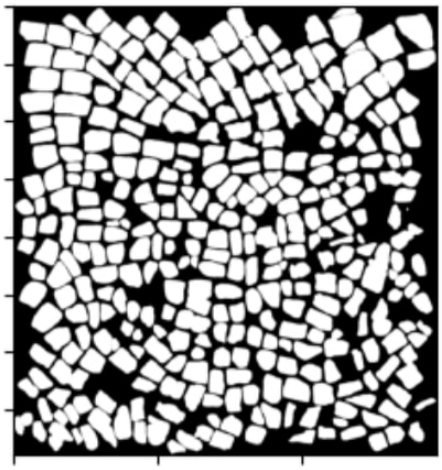
(a) Image originale

(b) Génération des masques sans recouvrement

FIGURE 1 – Segmentation sans recouvrement (tuilage 3x2)

Cependant, avec cette méthode, on observe sur la figure ci-dessus que les masques sont tronqués aux intersections des tuiles. Il est donc nécessaire de mettre en place un recouvrement des tuiles pour garantir que la forme des masques aux intersections soit correcte.

(a) Choix d'une taille pour le recouvrement



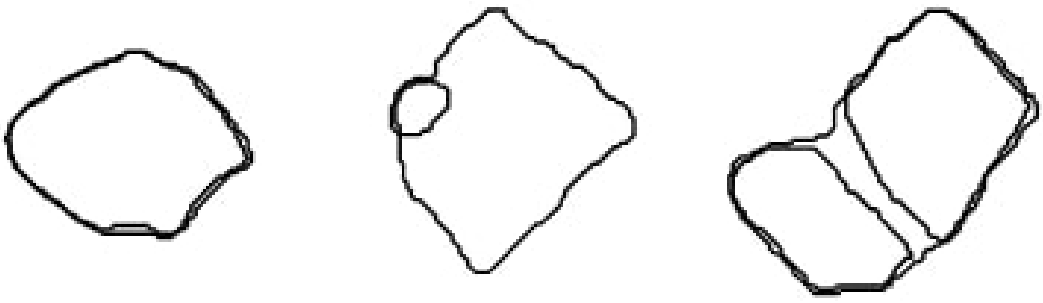
(b) Suppressions des masques sur les bords

Pour déterminer la taille du recouvrement, l'utilisateur choisit une taille standard en encadrant une grande tesselle. On multiplie ensuite cette taille par 1,25 pour obtenir la dimension du recouvrement (a).

Pour éviter d'ajouter des doublons dans les masques, les tesselles qui touchent les bords des tuiles lors du recouvrement sont supprimées (b).

2.4 Post-traitement des masques

Un post-traitement des masques générés est bien sûr nécessaire, car plusieurs cas de figure peuvent poser problème. Tout d'abord, il faut s'assurer que la taille d'un masque ne soit pas aberrante par rapport aux autres masques d'une même tuile. Pour cela, on détecte les données aberrantes statistiquement en fonction de la moyenne et de l'écart type, puis on supprime les masques correspondant à des objets reconnus par SAM ou ceux qui sont trop petits, car ils représentent souvent des fragments de tesselles ou d'autres imperfections.
Ensuite, on applique une opération morphologique de fermeture pour éliminer toute poussière résiduelle. Enfin, il est nécessaire de traiter trois types de masques, illustrés sur la figure ci-dessous.



(a) Doublons (b) Faible recouvrement (c) Recouvrement partiel

FIGURE 3 – Exemple de masques à supprimer

Pour ce faire, pour chaque tuile, on procède comme suit : on trie la liste des masques par taille croissante. Ensuite, on créer un masque vide de la taille de la tuile, qui servira de masque

intermédiaire. On va progressivement intégrer les masques, en commençant par les plus petits, sur ce masque intermédiaire.

Pour chaque masque courant, on calcul la similarité entre ce masque et l'intersection de celui-ci avec le masque intermédiaire. La similarité entre deux masques M_1 et M_2 est déterminée par la formule suivante :

$$\text{Similarité}(M_1, M_2) = \frac{\sum_{x \in M_1 \cap M_2} x}{\sum_{x \in M_1 \cup M_2} x}$$

Cette mesure permet de détecter les doublons (a) en appliquant un seuil. De plus, certains masques peuvent être le résultat de fragments de tesselles ou de textures non homogènes (b). Si un masque est entièrement contenu dans un autre et recouvre moins de 20% du masque plus grand, on supprime le masque plus petit.
Ce seuil de 20% est choisi pour éviter la suppression incorrecte de tesselles, comme illustré dans l'exemple (c), où le masque plus grand doit être conservé même s'il recouvre complètement le plus petit. Pour ces cas particuliers, un masque est supprimé s'il est recouvert à plus de 80% par d'autres masques déjà intégrés.

En option, j'ai ajouté une étape pour réduire la fréquence des cas illustrés dans la figure (c). En effet, le seuil de 20% peut être trop restrictif et entraîner la perte de certaines tesselles si elles sont combinées avec des tesselles voisines plus grandes. Pour limiter les erreurs dans de telles configurations, j'ai implémenté une étape préliminaire.

Cette étape consiste à désassembler les masques contenant plus d'une tesselle et qui peuvent être séparés facilement par une opération morphologique, comme illustré ci-dessous. On applique ce processus qu'aux 10% des masques les plus grands parmi ceux générés. En effet, les masques contenant plusieurs tesselles sont souvent les masques les plus grands.

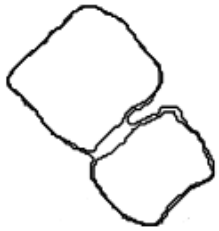


FIGURE 4 – Masque recouvrant facilement détachable

Pour détacher les masques contenant plusieurs tesselles, on applique une opération morphologique de fermeture dans une boucle `while`, en augmentant progressivement le nombre d'itérations de cette opération pour séparer les masques. Ensuite, on détecte le nombre de parties connexes dans l'image.

Si deux parties connexes sont détectées, on utilise les centroïdes de ces parties pour reconstruire progressivement les deux masques distincts. Ce processus consiste à modifier l'élément structurant et à appliquer des opérations d'ouverture dans la zone du masque original pour séparer les tesselles.

La figure ci-dessous illustre un exemple de séparation d'un masque contenant deux tesselles."

(a) Masque détachable

(b) Masques après séparation

FIGURE 5 – Séparation d'un masque à deux tesselles en deux masques distincts

Après ce traitement, les masques sont correctement positionnés, mais il reste des doublons dus au tuilage. Pour les éliminer, il suffit de calculer la similarité entre les masques lorsque l'on les intègre tuile par tuile. Cette approche permet de supprimer les doublons et d'obtenir un résultat correct comme cela çi-dessous.

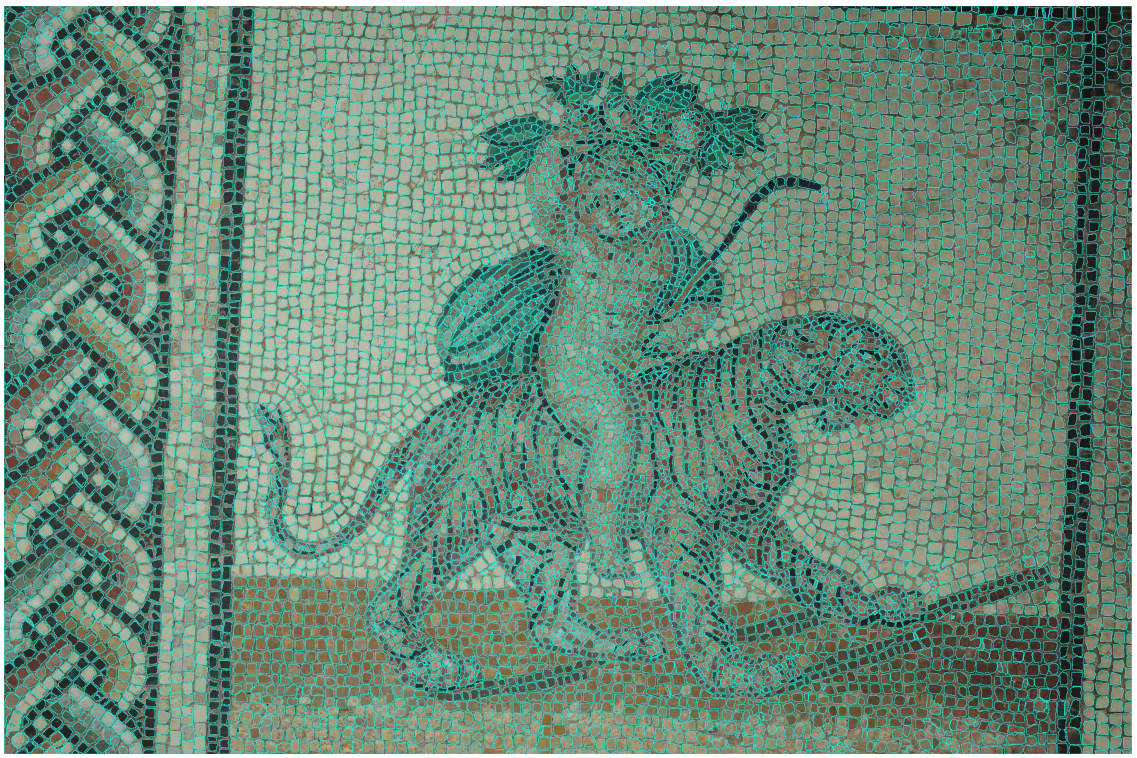


FIGURE 6 – Superposition des masques sur l'image originale

3 Exploitation de la segmentation

3.1 Caractéristiques des tesselles

Grâce à la segmentation, j'ai pu identifier toutes les tesselles détectées et obtenir des informations sur leurs tailles et leurs couleurs. De plus, en utilisant une image de normales, j'ai calculé un indice de rugosité. Pour ce faire, on calcul la variance des vecteurs de normales pour chaque composante (x, y, z). La variance mesure la dispersion des normales autour de la moyenne, ce qui reflète les variations de la surface. L'indice de rugosité est ensuite obtenu en faisant la moyenne des variances calculées pour chaque composante. Plus cet indice est élevé, plus la surface est rugueuse.

3.2 Regroupement des tesselles

En utilisant les caractéristiques des tesselles, j'ai pu les regrouper selon leur taille ou leur couleur en fonction de la taille moyenne des tesselles et d'une couleur d'intérêt sélectionnée par l'utilisateur. Pour identifier les couleurs similaires à la couleur d'intérêt, on utilise la distance euclidienne entre les couleurs RGB et appliquons un seuillage.
J'ai également réalisé un partitionnement automatique des tesselles en quatre clusters basés sur leurs couleurs. Pour ce faire, j'ai utilisé l'algorithme k-means pour le clustering.

4 Améliorations possibles

Malgré le traitement effectué, certains masques peuvent être omis, en particulier si les tesselles sont très foncées. Une amélioration significative de la segmentation des tesselles pourrait être obtenue en fine-tunant le modèle. Actuellement, je ne disposais pas d'une base de données avec des segmentations préalables sur des exemples de mosaïques de référence. Avec une telle base de données, il serait possible de modifier les poids du modèle SAM pour les adapter spécifiquement à notre problème, c'est-à-dire l'identification des tesselles dans une mosaïque. Cela permettrait de simplifier les étapes intermédiaires du traitement des images.

J'ai également exploré la possibilité de resegmenter une zone de l'image si elle n'a pas été correctement segmentée. Cependant, cette méthode ne s'est pas révélée efficace lorsque l'image reste inchangée. De plus, il serait souhaitable d'ajouter un outil permettant de compléter manuellement les tesselles manquantes, si nécessaire.

5 Interface graphique

Toutes les instructions d'utilisation de l'interface graphique sont fournies dans le manuel d'utilisation. L'interface graphique a été réalisée en utilisant les bibliothèques tkinter et customtkinter en Python.

6 References

Références

[1] Lamia Benyoussef & Stéphane Derrode, *Tessella-oriented segmentation and guidelines estimation of ancient mosaic images.*

[2] Qinpeng Guo, Yuchen Wang, Shijiao Yang & Zhibin Xiang, *A method of blasted rock image segmentation based on improved watershed algorithm.*

[3] Thibaud Ehret & Jean-Michel Morel, *Line Segment Detection : a Review of the 2022 State of the Art.*