ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

RAPPORT TECHNIQUE PRÉSENTÉ À MONSIEUR LUC DUONG DANS LE CADRE DU COURS COMPRÉHENSION DE l'IMAGE

LABORATOIRE 2

PAR
KATIA KACI

MONTRÉAL, LE 17 FÉVRIER 2025

Table des matières

Introduction

La grande majorité des personnes utilisent désormais leur téléphone cellulaire pour prendre des photos lors d'événements marquants ou pendant leurs voyages. Ces appareils offrent une grande simplicité d'utilisation grâce à de nombreuses fonctionnalités, telles que l'autofocus, le zoom ou encore la possibilité de facilement prendre des photos panoramiques. Les photos panoramiques permettent de fusionner plusieurs images prises l'une après l'autre afin de créer une seule image offrant une vue étendue d'un paysage ou d'une scène. Cette technique peut être particulièrement utile pour capturer l'ampleur d'un site touristique, d'un paysage naturel ou d'une scène urbaine, en donnant une impression de continuité et de profondeur, ce qui serait difficile à capturer avec une photographie standard. L'objectif de ce deuxième laboratoire est de développer une application simple permettant de fusionner des images prises sous différents angles en un seul panorama, tout comme la fonctionnalité des téléphones intelligents. Pour ce faire, nous nous sommes inspirés d'un tutoriel en ligne sur la fusion d'images avec OpenCV et Python, réalisé par Adrian Rosebrock (A. Rosebrock, 2018). L'approche que nous avons développée se divise en deux grandes étapes. La fusion des images pour créer le panorama et le recadrage afin d'obtenir une image bien cadrée, sans contours noirs. La fusion repose sur des techniques d'alignement d'images basées sur la méthode proposée par M. Brown et D. G. Lowe dans leur article *Automatic Panoramic Image Stitching using Invariant Features*, publié en 2007, puis le recadrage consiste à éliminer les zones indésirables pour mettre en valeur l'ensemble de l'image finale, tout comme un téléphone intelligent le ferait. Ce rapport présentera dans un premier temps une description détaillée de cette approche, en expliquant les différentes étapes techniques. Par la suite, nous analyserons et discuterons des résultats obtenus, en présentant les limites de notre implémentation.

Approche proposée

En ce qui concerne la fusion d'images, nous utilisons les fonctionnalités de fusion d'OpenCV, notamment *cv2.createStitcher*, *cv2.Stitcher_create* et *stitch(images)*. L'intégration de ces fonctionnalités dans Python est très simple : il suffit de lire nos images, de les ajouter à une liste, d'initialiser le fusionneur (*stitcher*) d'OpenCV et de lui transmettre nos images à l'aide de la fonction *stitcher.stitch(images)*. L'implémentation de cette fonctionnalité n'est donc pas très complexe en Python. Toutefois, l'algorithme sur lequel se base l'implémentation d'OpenCV est basé sur la méthode proposée par M. Brown et D. G. Lowe dans leur article *Automatic Panoramic Image Stitching using Invariant Features*, publié en 2007. Dans un premier lieu, l'algorithme reconnait et extrait des points d'intérêt dans l'image grâce à l'algorithme SIFT. Ces points, que l'on va appeler ici des caractéristiques, sont des caractéristiques qui sont faciles à reconnaitre et qui restent stables d'une image à l'autre, et ce, malgré des changements d'éclairage ou des modifications lors de la prise de la photo (par exemple, changement d'angle). Par la suite, des correspondances sont établies entre ces caractéristiques dans les différentes images. Pour créer ces correspondances, l'algorithme de M. Brown et D. G. Lowe utilise la méthode RANSAC afin de sélectionner un ensemble de caractéristiques compatibles avec une homographie entre les images. Ensuite, un modèle probabiliste est appliqué pour vérifier ces correspondances. Ces étapes permettent alors d'aligner les images entre elles. Par la suite, une étape d'optimisation est réalisée afin de permettre d'améliorer la cohérence de l'alignement des images tout en diminuant les erreurs accumulées, ce qui donne une fusion plus précise et homogène du panorama. Cette étape est appelée dans le *Bundle Adjustment* dans l'article de M. Brown et D. G. Lowe. Après cet ajustement, une phase de redressement automatique du panorama est effectuée afin de s'assurer d'obtenir une image globalement horizontale. Finalement, une étape d'ajustement du gain entre les images, suivi d'un *Multi-Band Blending*, est appliquée pour combiner parfaitement les images et créer l'illusion qu'il s'agit d'une seule image panoramique (Brown et Lowe, 2007). Ces étapes nous permettent alors d'obtenir une image comme celle de la figure 1.

Figure 1. Résultat final de la fusion panorama de 3 images à l'aide des fonctionnalités OpenCV basées sur la méthode proposée par M. Brown et D. G. Lowe

En ce qui concerne le recadrage de l'image, le défi consiste à déterminer comment couper le panorama de manière à perdre le moins de détails possible tout en éliminant toutes les bordures noires. Cette étape a été réalisée dans le but de s'approcher au maximum d'une image finale qu'une solution commerciale pourrait offrir. Pour implémenter ce recadrage, nous nous sommes encore une fois inspirés du tutoriel d'Adrian Rosebrock (A. Rosebrock, 2018). Nous commençons par ajouter une bordure de 10 pixels sur tous les côtés de l'image afin de nous assurer qu'une bordure noire apparaisse dans le panorama. Ensuite, nous créons une copie de notre image panoramique en niveaux de gris (*Gray Scale*) au lieu de RGB, puis une autre copie en appliquant un seuil binaire (*threshold*). Cela signifie que tous les pixels noirs de l'image deviennent 0 et tous les pixels gris (peu importe la tonalité) passent à 255, apparaissant alors en blanc. La figure 2 illustre ces copies d'images et leurs modifications.

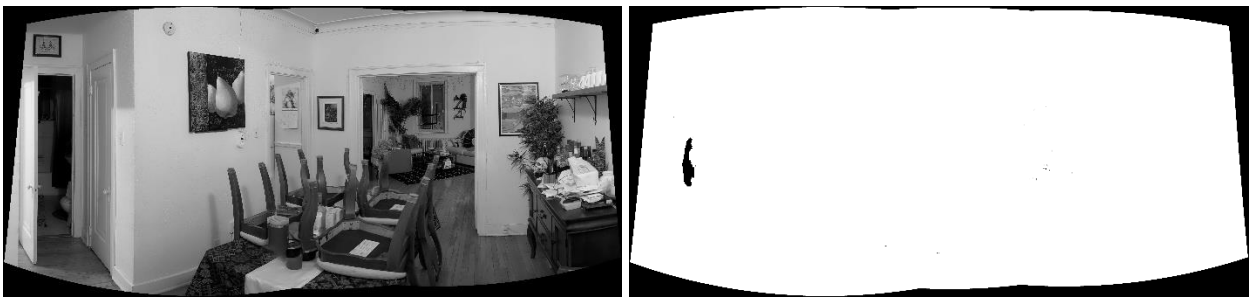

Figure 2. Copies de l'image panorama en *Gray Scale* (gauche) et en *Threshold* Binaire (droite).

Cependant, ce procédé posait un problème. Lorsque l'image contenait des pixels noirs à l'intérieur du panorama, on se retrouvait non seulement avec des contours noirs, mais aussi avec des zones internes noires. Ceci causait des erreurs avec la méthode implémentée dans le tutoriel et le recadrage de l'image ne se faisait pas bien. Pour résoudre ce problème, nous avons ajouté une étape supplémentaire non présente dans le tutoriel, qui nous permet d'identifier tous les contours noirs internes afin de bien faire fonctionner la solution présentée dans le tutoriel pour le recadrage. Pour ce faire, nous utilisons la fonction d'OpenCV *cv2.findContours()* en différenciant les contours externes des contours internes à l'aide d'une hiérarchie avec *cv2.RETR_CCOMP*, ce qui nous permet de déterminer quels contours sont internes. Ensuite, nous créons un masque pour les zones noires internes et dessinons uniquement ces pixels en blanc. Enfin, nous combinons l'image des zones internes traitées avec celle obtenue par le *threshold*. Ces étapes sont illustrées dans la figure 3. Par la suite, nous poursuivons avec les étapes du tutoriel, qui consistent à créer une *Bounding Box* à l'aide des contours, qui sont désormais les seuls pixels noirs de l'image *threshold*. Cette *Bounding Box* se réduit alors progressivement jusqu'à ce qu'elle puisse s'insérer dans la partie intérieure du panorama, sans inclure de pixels noirs issus des contours extérieurs. Enfin, nous coupons l'image à l'aide des coordonnées de la *Bounding Box* pour obtenir l'image finale (A. Rosebrock, 2018), présentée à la figure 4.

Figure 3. Processus de détection des pixels noirs à l'intérieur du panorama. 1) Image *Threshold* Binaire (image de gauche). 2) Détection et mise en blanc des pixels des contours intérieurs (image au centre). 3) Combinaison des deux images (image de droite).

Figure 4. Résultat final de la fusion panorama de 3 images et du recadrage de l'image.

Une fois l'application fonctionnelle, nous avons analysé les limites de notre implémentation en effectuant la fusion d'images de plusieurs images recueillies. Notre objectif est de vérifier quelles images fonctionnent bien et lesquelles fonctionnent moins bien. Nous avons donc utilisé plusieurs jeux d'images prises sous différentes configurations (perspective, éclairage, etc.) afin d'analyser notre algorithme face à des conditions variées.

Résultats et discussion

Notre application nous permet d'obtenir d'excellents résultats dans l'ensemble. Cependant, elle comporte tout de même certaines limites.

Tout d'abord, lorsque le chevauchement entre images est faible, c'est-à-dire lorsque les images ne partagent pas un assez grand champ de vision commun avec les images voisines, l'algorithme n'arrive pas à détecter les caractéristiques communes entre les images, et elles ne sont alors pas fusionnées, ce qui fait en sorte que notre application ne parvient pas à générer la photo panoramique. Le chevauchement des images doit être d'environ 15% à 40% pour garantir un bon alignement (Adobe, 2022).

Dans l'ensemble d'images de la figure 5 ci-dessous, nous pouvons constater que le chevauchement entre images est assez minime. Par exemple, seuls le coin du mur et la petite prise électrique pourraient potentiellement être considérés comme des points d'intérêts communs entre la première et la deuxième image. Or, ces éléments sont peu distinctifs, d'autant plus qu'ils sont entourés d'un mur blanc et d'un sol assez uniforme. Entre la deuxième et la troisième image, le problème est encore plus marqué. Le mur droit visible dans la deuxième image est très peu visible dans la troisième et est beaucoup plus sombre, ce qui réduit le nombre d'éléments communs que peut utiliser l'algorithme. Le couloir en perspective dans la deuxième image se prolonge dans la troisième, mais la grande fenêtre introduit une certaine discontinuité, puisque la scène extérieure visible à travers la fenêtre introduit des éléments en arrière-plan, et ce type de variation peut entrainer un effet de parallaxe qui peut fausser l'alignement entre les images (M. McGuffin et P. Robitaille, communication personnelle, 2025). Ainsi, dû à ce manque de chevauchement, notre algorithme n'est pas en mesure de détecter un nombre suffisant de points caractéristiques correspondants, ce qui empêche la génération du panorama de ces images.

Figure 5. Jeu d'images dont il est impossible de créer une fusion panorama

Un autre problème probable est l'angle de prise de vue. Lorsque les images ont des perspectives trop différentes, les points caractéristiques communs sont plus difficiles à déterminer. Ces différences de perspectives peuvent être dues à la hauteur et à l'orientation de l'appareil au moment de la capture des photos, qui peuvent varier si la personne n'est pas parfaitement stable. Pour contrer ce problème, l'utilisation d'un trépied ou d'un support pour appareil photo pourrait être envisagée.

De plus, notre application ne performe pas adéquatement lorsque les images sont des scènes qui manquent de détails ou qui présentent de grandes surfaces uniformes (par exemple, un ciel bleu sans nuages, une scène enneigée, ou encore un mur blanc ou un sol d'une couleur unie comme sur l'exemple précédent). En effet, la présence de zones homogènes réduit le nombre de points clés détectables. Les algorithmes de *stitching*, en particulier ceux reposant sur des détecteurs de *features* comme SIFT, fonctionnent mieux avec des images qui contiennent des textures riches et variées. L'ensemble d'images présenté dans la figure 6 ci-dessous, qui ne nous permet pas de générer un panorama complet, nous a permis d'identifier cette limite de notre application.

Figure 6. Jeu d'images dont la fusion panorama est incomplète

Figure 7. Résultat obtenu pour l'ensemble d'images de la figure 6 avec notre application

Bien que les images initiales partagent quelques points communs (l'arbre, le toit de la maison, les câbles électriques), le manque de points caractéristiques distincts dans la grande zone bleue uniforme du ciel rend l'assemblage des images complexe. L'absence de structures ou de motifs, comme des bâtiments, des nuages ou des objets en premier plan, réduit également l'efficacité du modèle. Le résultat obtenu avec notre application est visible dans la figure 7, où l'on peut observer une fusion partielle du panorama (images 1 et 2 seulement), avec des discontinuités et des alignements imparfaits (voir les câbles électriques en bas de l'image).

Pour mieux comprendre les limites de notre approche, nous avons testé le même jeu d'images sur un outil en ligne de création de panoramas. Comme le montre la figure 8, cet outil a permis d'obtenir une meilleure fusion, probablement grâce à une approche différente de la nôtre pour l'assemblage. Cette comparaison met en évidence les limites de notre application et nous indique qu'il faut envisager des pistes d'amélioration, notamment en ce qui concerne la gestion des zones avec peu ou sans texture et l'optimisation du recadrage des images.

Figure 8. Résultat obtenu pour l'ensemble d'images de la figure 6 avec *Sisik* (https://sisik.eu/pano)

Hormis les cas d'utilisation spécifiques mentionnés plus haut, notre application présente d'excellents résultats. En voici quelques-uns à titre d'exemples.

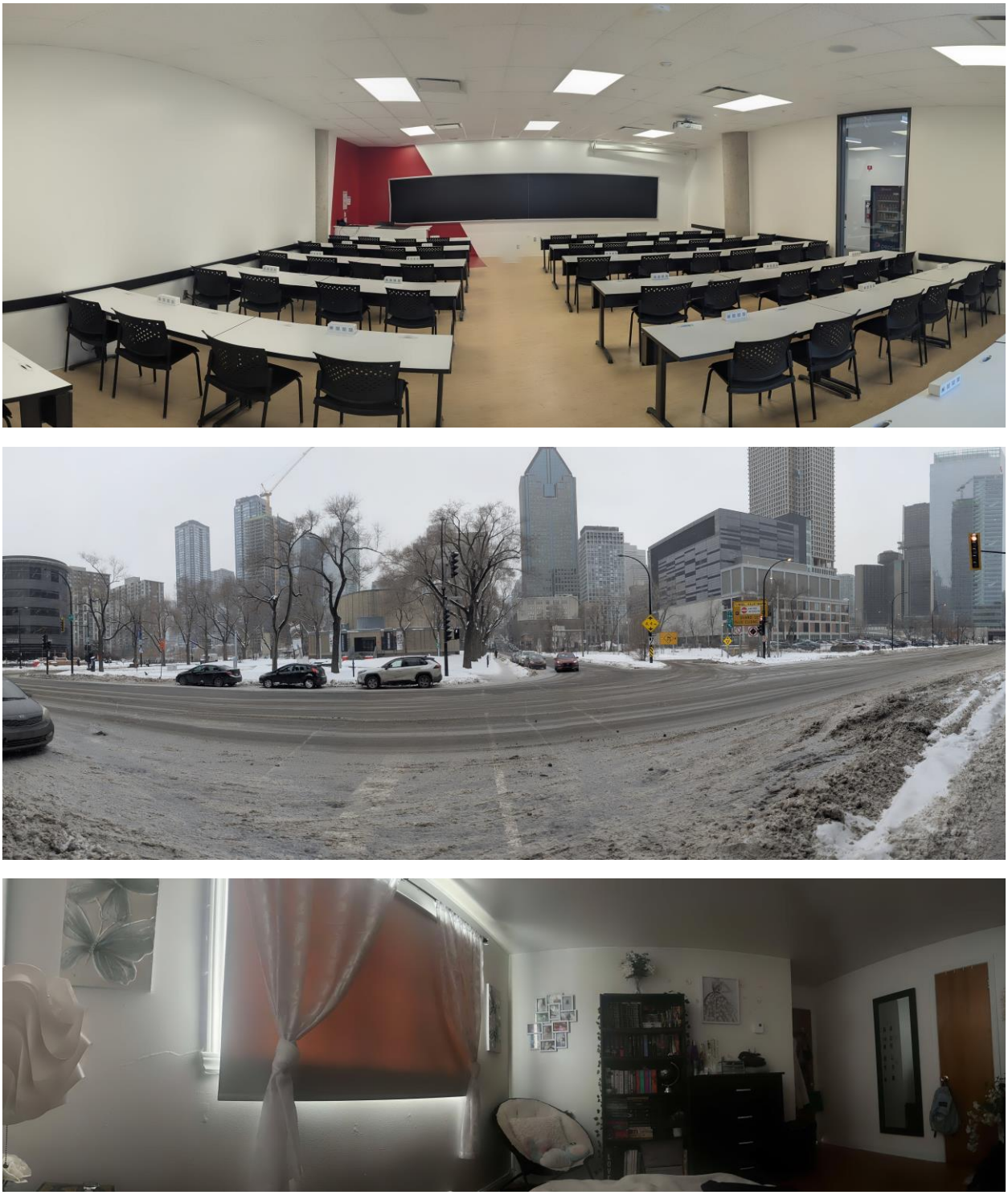

Figure 9. Résultats obtenus pour différents ensembles d'images

Conclusion

Ce laboratoire nous a permis d'explorer et d'implémenter un algorithme de fusion d'images pour la création de panoramas en nous basant sur les techniques d'alignement et d'assemblage proposées par M. Brown et D. G. Lowe. En utilisant les fonctionnalités d'OpenCV, nous avons développé une application capable de fusionner un ensemble d'images en un seul panorama, puis d'appliquer un recadrage automatique afin d'éliminer les bordures noires.

Les résultats obtenus démontrent que notre approche fonctionne très bien dans la majorité des cas, à condition que les images aient un chevauchement suffisant, que les angles de prise de vue ne varient pas trop, et que les images contiennent assez de texture et de détails distinctifs. Ces limites peuvent entraîner des images panoramiques imparfaites, incomplètes ou tout simplement l'incapacité de générer un panorama.

L'efficacité de notre application dépend également des paramètres utilisés. Par exemple, le choix du détecteur de caractéristiques influence directement la robustesse de l'alignement de notre algorithme. Dans notre cas, nous avons utilisé SIFT, puisqu'il est reconnu pour sa précision et sa robustesse face aux changements d'éclairage et de perspective. Il serait toutefois intéressant d'explorer d'autres détecteurs, comme ORB, qui est plus rapide et qui est souvent reconnu comme étant la meilleure méthode d'extraction de caractéristiques des deux. (Kennerley, M., 2021).

En conclusion, ce laboratoire nous a permis de mieux comprendre l'importance des conditions de prise de vue, les défis liés à la fusion d'images et de nous familiariser à l'implémentation d'algorithmes de vision par ordinateur. 1qaz

Bibliographie

Rosebrock, A. (2018, 17 December). Pyimagesearch_Sales_page w/out Autoplay. PyImageSearch. Repéré à https://pyimagesearch.com/2018/12/17/image-stitching-with-opencv-and-python/

Brown, M., & Lowe, D. G. (2006). Automatic Panoramic Image Stitching using Invariant Features. International Journal of Computer Vision, 74(1), 59–73. https://doi.org/10.1007/s11263-006-0002-3

Création d'un panorama. (s.d.). Repéré à https://helpx.adobe.com/ca_fr/photoshop-elements/using/stitching-together-panoramas.html

Keypoint detector. (s.d.). Repéré à https://www.cs.ubc.ca/~lowe/keypoints/

Kennerley, M. (2021, 21 May). A Comparison of SIFT, SURF and ORB on OpenCV - Mikhail Kennerley. Medium. Repéré à https://mikhail-kennerley.medium.com/a-comparison-of-sift-surf-and-orb-on-opencv-59119b9ec3d0