Rapport: Assemblage d'Images par Transformation Homographique

Votre Nom

October 31, 2024

Contents

1 Introduction

Ce rapport décrit un programme d'assemblage d'images, qui permet de réorganiser et d'aligner plusieurs images en utilisant des transformations homographiques. Cette application a été développée avec la bibliothèque OpenCV en C++. Le but est de sélectionner des points correspondants entre les images, de calculer la transformation homographique, puis de fusionner les images pour obtenir un panorama.

2 Description du Code

Le code est divisé en plusieurs parties pour effectuer les étapes de chargement, réorganisation, sélection de points, alignement, et assemblage des images.

2.1 Chargement des Images

Les images sont chargées à partir des chemins spécifiés dans le vecteur `imagePaths`. Elles sont ensuite stockées dans un vecteur nommé `images` pour un accès facile tout au long du programme.

```
vector<string> imagePaths = {"DVI1/gi05.jpeg", "DVI1/gi02.jpeg"};
for (const auto& path : imagePaths) {
    Mat img = imread(path);
    if (img.empty()) {
        cout << "Could not open image: " << path << endl;
        return -1;
    }
    images.push_back(img);
}
```

2.2 Réorganisation des Images

Une fenêtre nommée "Image Order" est affichée pour permettre la réorganisation des images. La fonction `swapImages` prend deux indices et échange les images correspondantes. La fonction de rappel de souris `onMouseReorder` gère les clics de l'utilisateur pour choisir deux images à échanger.

```
void swapImages(int idx1, int idx2) {
    if (idx1 >= 0 && idx1 < images.size() && idx2 >= 0 && idx2 < images.size())
        Mat temp = images[idx1];
        images[idx1] = images[idx2];
```

```
        images[idx2] = temp;
    }
}
```

2.3 Sélection des Points Correspondants

Pour chaque paire d'images, l'utilisateur doit sélectionner quatre points correspondants dans l'image de base et l'image actuelle pour calculer la transformation homographique. La fonction de rappel de souris `onMouseSelectPoints` permet de stocker les coordonnées des points sélectionnés par l'utilisateur dans deux vecteurs, `points1` et `points2`.

```
void onMouseSelectPoints(int event, int x, int y, int, void* userdata) {
    if (event == EVENT_LBUTTONDOWN) {
        if (points1.size() < 4) {
            points1.push_back(Point2f(x, y));
            cout << "Point for base image: (" << x << ", " << y << ")" << endl;
        } else if (points2.size() < 4) {
            points2.push_back(Point2f(x, y));
            cout << "Point for current image: (" << x << ", " << y << ")" << end
        }
    }
}
```

2.4 Calcul et Application de la Transformation Homographique

Une fois les points correspondants sélectionnés, la transformation homographique est calculée avec la fonction `findHomography`. Cette transformation permet d'aligner l'image suivante avec l'image de base, en utilisant la méthode RANSAC pour minimiser les erreurs dues aux correspondances incorrectes.

```
Mat H = findHomography(points2, points1, RANSAC);
warpPerspective(img2, warped, H, Size(result.cols + img2.cols, result.rows));
```

2.5 Fusion des Images

Après avoir appliqué la transformation, l'image transformée est fusionnée avec l'image de base pour créer un panorama. La fusion est réalisée en copiant les pixels de l'image transformée sur l'image de résultat.

```
Mat temp(Size(result.cols + warped.cols, result.rows), result.type());
result.copyTo(temp(Rect(0, 0, result.cols, result.rows)));
warped.copyTo(temp(Rect(result.cols, 0, warped.cols, warped.rows)));
result = temp.clone();
```

3 Théorie de la Transformation Homographique

La transformation homographique permet de mapper des points d'une image vers une autre lorsque les deux images sont prises d'un point de vue différent mais sur un plan commun. Mathématiquement, la transformation homographique est une matrice H de taille 3×3 qui transforme un point $p = (x, y)$ d'une image source en un point $p' = (x', y')$ dans l'image cible.

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = H \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{1}$$

où H est calculée par une méthode basée sur les moindres carrés, souvent en utilisant l'algorithme RANSAC pour éliminer les correspondances incorrectes.

4 Étapes du Processus

4.1 Réorganisation des Images

L'utilisateur clique sur deux images pour échanger leur position dans l'ordre d'affichage. Ceci est effectué en utilisant une fonction de rappel de souris pour capturer les clics de l'utilisateur.

4.2 Sélection des Points Correspondants

Pour chaque paire d'images, l'utilisateur sélectionne quatre points d'intérêt dans les deux images. Ces points sont utilisés pour calculer la matrice de transformation homographique.

4.3 Alignement et Fusion des Images

La transformation homographique est appliquée à l'image suivante, qui est ensuite fusionnée avec l'image de base pour obtenir un panorama complet. Ce processus est répété pour toutes les images.

5 Résultats

Les résultats finaux montrent les images dans leur état initial et l'image assemblée finale.

5.1 Images d'État Initial

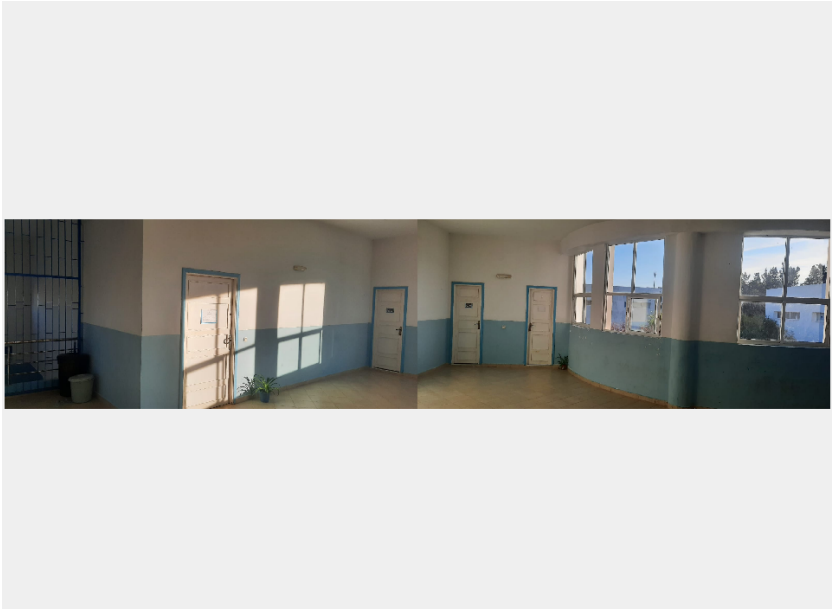

Figure 1: État initial des images

5.2 Image Assemblée Finale

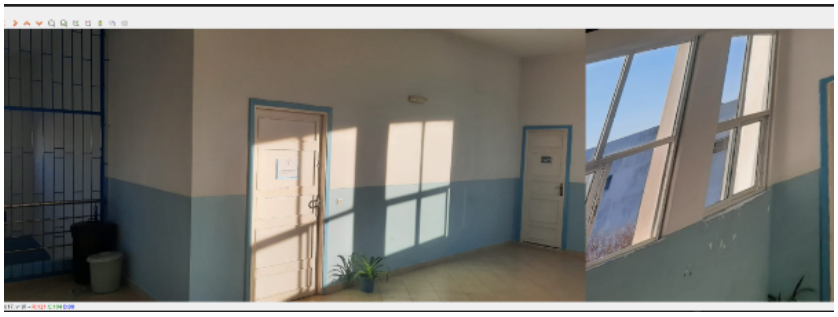

Figure 2: Image finale assemblée

6 Conclusion

Le programme a démontré la capacité de créer un panorama en utilisant des transformations homographiques basées sur des points correspondants sélectionnés par l'utilisateur. Bien que la méthode demande une intervention manuelle pour la sélection des points, elle fournit des résultats précis pour des scènes en plan. Pour une automatisation complète, des techniques de correspondance de points automatiques pourraient être envisagées.