TP – Construction d'une mosaïque à partir de plusieurs images

17 octobre 2024

Objectifs

Le but de ces quatre séances de travaux pratiques est de réaliser un programme, en `matlab`, permettant de créer une mosaïque d'images à partir de plusieurs images, au minimum deux, qui possèdent une zone de recouvrement ou zone commune, cf. figure 1.

FIGURE 1 – (a) Images (en couleur) utilisées et (b) mosaïque construite à partir des 2 images (les deux premières).

Données fournies

- `filet1.pgm`, `filet2.pgm`, `filet3.pgm` : trois images pour tester les fonctions. Nous donnons également une version en couleur dans `filet*.jpg`.
- `harris.m` : permettant de réaliser la détection de points d'intérêt selon Harris, **à compléter**.
- `apparier_POI.m`, `voisinage.m` : fonctions qui correspondent respectivement à la mise en correspondance par corrélation et à la détermination du voisinage pris en compte lors de la recherche d'un correspondant, **à compléter**.
- `apparier_Points.m` : fonction permettant d'appliquer des contraintes sur la mise en correspondance, **à modifier également**.
- `affichage_image, affichage_POI, plot_points.m, affichage_appariement` : fonctions qui permettent d'afficher respectivement une image, des points d'intérêt sur une image et des appariements entre deux images (les numéros identiques indiquent les points qui se correspondent).
- `TP_mosaique.m` : script permettant d'effectuer les étapes nécessaires de la détection à la construction de la mosaïque. **Certaines parties sont commentées en attendant de compléter les fonctions correspondantes.**
- `homographie.m`, `appliquerHomographie.m` : **fonctions à compléter**. La première permet d'estimer une homographie alors que la seconde permet de l'appliquer à un ensemble de points.
- `mosaique.m` : fonction permettant la construction de la mosaïque, une fois que l'homographie a été estimée. Cette fonction devra être **copiée et modifiée**.

Travail à faire pour le 14 novembre 2024

Vous devez travailler **en binôme** sur ce TP et **répondre individuellement à un questionnaire sur moodle le 14 novembre de 11h30 à 12h**, à la fin de la dernière séance de TP. De plus, à la fin des quatre séances de TP, vous devez déposer sur `moodle`, **au plus tard le 14 novembre 2024 à 18h30**, une archive de nom `Nom1_Nom2.zip` contenant **uniquement** les fichiers et dossier suivants :

- `harris.m`, fonction relative à la détection ;
- `voisinage.m`, `apparier_POI.m`, `apparier_Points.m`, fonctions relatives à l'appariement ;
- `homographie.m`, `mosaique.m`, `mosaiquebis.m`, `mosaiquecoul.m` et `mosaique3.m`, fonctions relatives à la mosaïque ;
- `TP_mosaique.m`, le script fourni pour l'exécution de toute la chaîne de traitement et compléter pour faire fonctionner la mosaïque à 3 (voire N) images. Idéalement, ce serait bien d'avoir la possibilité de générer tous les résultats que vous avez pu obtenir sur toutes les images que vous aurez réussi à traiter.
- `Images`, un dossier contenant toutes les images de tests que vous avez ajoutées et les images des mosaïques obtenues.

Enfin, sous moodle, **vous devrez répondre à une activité de sondage pour choisir un créneau de passage pour la démonstration de votre code**.

1 Démarche proposée

Nous supposons que l'alignement géométrique des images peut s'effectuer selon une homographie [1]. Cette hypothèse peut être faite lorsque les images sont prises avec une caméra à laquelle une rotation autour de son centre optique a été appliquée. La démarche proposée est la suivante :

(1) Détecter des points d'intérêt avec le détecteur de Harris.

(2) Mettre en correspondance ces points par une approche par corrélation.

(3) Estimer l'homographie à partir des correspondances obtenues en (2).

(4) Construire une mosaïque en utilisant l'homographie estimée en (3).

Remarque importante Pour commencer, le travail s'effectuera avec les images en niveau de gris.

2 Détection de points d'intérêt par l'opérateur de Harris

Cette partie correspond aux fonctions suivantes :

- `plotpoints.m` : fonction qui permet d'afficher des points d'intérêt sur une image ;
- `harris.m` : fonction qui détermine les points d'intérêt dans l'image.

2.1 Rappel

Soit I l'image étudiée, et nous posons :

$$A = w * \left(\frac{\partial I}{\partial x}\right)^2, \quad B = w * \left(\frac{\partial I}{\partial y}\right)^2 \quad \text{et} \quad C = w * \left(\frac{\partial I}{\partial x}\frac{\partial I}{\partial y}\right)$$

1. Une homographie est une transformation linéaire entre deux plans projectif. Dans la suite du TP, nous reprendrons cette notion.

où w est un filtre gaussien (de support carré, la plupart du temps) et $*$ l'opérateur de convolution. Nous obtenons la réponse du détecteur de Harris en chaque point (x, y), en calculant :

$$R(x,y) = \mathrm{Det}(M(x,y)) - k\left(\mathrm{Trace}(M(x,y))\right)^2, \tag{1}$$

avec

- $M(x,y) = \begin{pmatrix} A(x,y) & C(x,y) \\ C(x,y) & B(x,y) \end{pmatrix}$,
- $\mathrm{Det}(M(x,y)) = A(x,y)B(x,y) - C(x,y)^2$ et
- $\mathrm{Trace}(M(x,y)) = A(x,y) + B(x,y)$.

Dans la littérature, il est conseillé de prendre k avec une valeur entre 0.04 et 0.06.

2.2 Calcul de la réponse R

Compléter la fonction `harris.m` en suivant les indications fournies dans la description suivante et le fichier correspondant.

Les paramètres d'entrée de cette fonction sont :

- `Im` : image pour laquelle les points d'intérêt sont extraits ;
- `TailleFenetre` : taille du filtre (nous supposons un filtre carré de taille `TailleFenetre` × `TailleFenetre` ;
- `NbPoints` : nombre de points sélectionnés.

Les paramètres de sortie de cette fonction sont :

- `XY` : coordonnées des points d'intérêt stockées dans une matrice de taille `NbPoints` × 2 ;
- `Res` : image des réponses maximales avant sélection des `NbPoints` points.

Pour écrire cette fonction, vous devez suivre ces étapes :

(1) *Calcul des dérivées images suivant les lignes et les colonnes* – Utiliser la fonction `gradient` ;

(2) *Calcul du filtre de lissage gaussien* – Utiliser la fonction `fspecial` ;

(3) *Calcul de la réponse R* – Utiliser l'équation (1) ;

(4) *Suppression des non-maxima locaux* – Une fois la réponse R calculée, il est nécessaire de ne garder que les maxima locaux. Pour cela, pour chaque point de l'image, il faut modifier la réponse, de la manière suivante :

$$R_{\text{modifie}}(x,y) = \begin{cases} R_{\max}(x,y) & \text{si } R(x,y) = R_{\max}(x,y) \\ 0 & \text{sinon.} \end{cases} \tag{2}$$

La réponse $R_{\max}(x,y)$ est la réponse maximale dans le voisinage `TailleFenetre` × `TailleFenetre` de (x,y).

(5) *Sélection des* `NbPoints` *points* – Une fois la réponse modifiée, il reste à sélectionner les points ayant les `NbPoints` réponses les plus fortes.

3 Appariement des points d'intérêt

Une fois l'étape de détection complétée, vous devez compléter les fichiers suivants :

- `voisinage.m` : fonction qui retourne les niveaux de gris dans le voisinage de chaque point d'intérêt dans une matrice (appelée par `apparier_Points.m`) ;
- `apparier_POI.m`, `apparier_Points.m` : fonctions permettant d'effectuer les appariements.

3.1 Rappel

Nous utiliserons l'algorithme de mise en correspondance par corrélation avec une stratégie de type *winner takes all* comme vu en cours.

Données
Images à traiter : I_1 et I_2
Point d'intérêt dans I_1, respectivement I_2 : $\mathbf{p}_1 = (x_1, y_1)$, $\mathbf{p}_2 = (x_2, y_2)$
Ensemble des points d'intérêt de l'image I_2 : $\texttt{PI}(I_2)$
Vecteur contenant les niveaux de gris dans le voisinage de $\mathbf{p}_1$, respectivement $\mathbf{p}_2$: $\mathbf{f}_1$, $\mathbf{f}_2$
Mesure de similarité entre deux vecteurs : $\texttt{Sim}(\mathbf{f}_1, \mathbf{f}_2)$
Algorithme de mise en correspondance

1. Pour chaque $\mathbf{p}_1 \in I_1$ faire

   Estimer $\mathbf{p}_2 \in I_2$ tel que $\mathbf{p}_2 = \underset{\mathbf{p} \in \texttt{PI}(I_2)}{\operatorname{argmin}} \texttt{Sim}(\mathbf{f}_1, \mathbf{f}_2)$

2. Ajout de contrainte (seuil, bidirectionnelle)

La mesure utilisée est la corrélation croisée centrée normalisée (*Zero mean Normalised Cross Correlation*) définie par :

$$\mathrm{ZNCC}(\mathbf{f}_1, \mathbf{f}_2) = \frac{(\mathbf{f}_1 - \overline{\mathbf{f}_1}) \cdot (\mathbf{f}_2 - \overline{\mathbf{f}_2})}{\|\mathbf{f}_1 - \overline{\mathbf{f}_1}\| \|\mathbf{f}_2 - \overline{\mathbf{f}_2}\|} \tag{3}$$

où $\overline{\mathbf{f}_i}, i = 1, 2$ est la moyenne des niveaux de gris des éléments de $\mathbf{f}_i$.

3.2 Description de la fonction voisinage

Les paramètres d'entrée de cette fonction sont :

- `I` : l'image à traiter ;
- `xyPt` : une matrice de taille `NbPoints`×2 contenant les coordonnées des points à apparier (il y a un point par ligne) ;
- `TailleFenetre` : la taille de la fenêtre de corrélation.

Le paramètre de sortie de cette fonction est `voisins` une matrice de taille `NbPoints`×(`TailleFenetre`)^2 contenant sur chaque ligne le voisinage du point d'intérêt correspondant.

Compléter la fonction `voisinage.m` en suivant les indications fournies dans le fichier correspondant.

Vérification

L'appel à `voisinage(Im1,[5 105; 125 235],3)` doit donner la réponse suivante :

```
ans =
 232   232   232   232   232   232   232   232   232
   158   171   161   159   169   165   165   160   162
```

3.3 Description de la fonction apparier_Points

La fonction `apparierPoints` permet de construire une matrice `C` de taille `NbPoints_1`×`NbPoints_2` avec `NbPoints_1`, respectivement `NbPoints_2`, le nombre de points d'intérêt détectés dans l'image I_1, respectivement I_2. Chaque élément `C(i,j)` de cette matrice correspond au score de corrélation entre les points d'intérêt d'indice `i` dans `Im1` et `j` dans `Im2`. Les points à apparier sont stockés dans `Ptint1` et `Ptint2`. Il faut également préciser la taille de la fenêtre de corrélation avec le paramètre d'entrée `K`.

Compléter la fonction `apparier_Points.m` en suivant les indications fournies dans le fichier.

Vérification

≪ Dé-commenter ≫ la ligne du script `TP_mosaique.m` qui correspond à l'appel de la fonction `apparierPOI` afin de réaliser l'appariement et vérifier visuellement les résultats.

3.4 Amélioration de la fonction apparierPOI

Les résultats précédents doivent contenir des erreurs d'appariement. Pour tenter de corriger ces erreurs, nous pouvons appliquer la contrainte de symétrie, cf. cours. Nous pouvons également appliquer un seuil sur les scores de corrélation obtenues afin de ne garder que les correspondances les plus fiables.

Modifier/compléter la fonction `apparier_POI.m` en suivant les indications fournies dans le fichier correspondant afin de mettre en œuvre ces deux améliorations. **Nous demandons d'effectuer ces améliorations en se plaçant uniquement du point de vue de l'image 1.**

Vérification

En prenant un seuil à 0.95 (déjà fixé dans `TP_mosaique`), vérifier que le résultat est similaire à la figure 2.

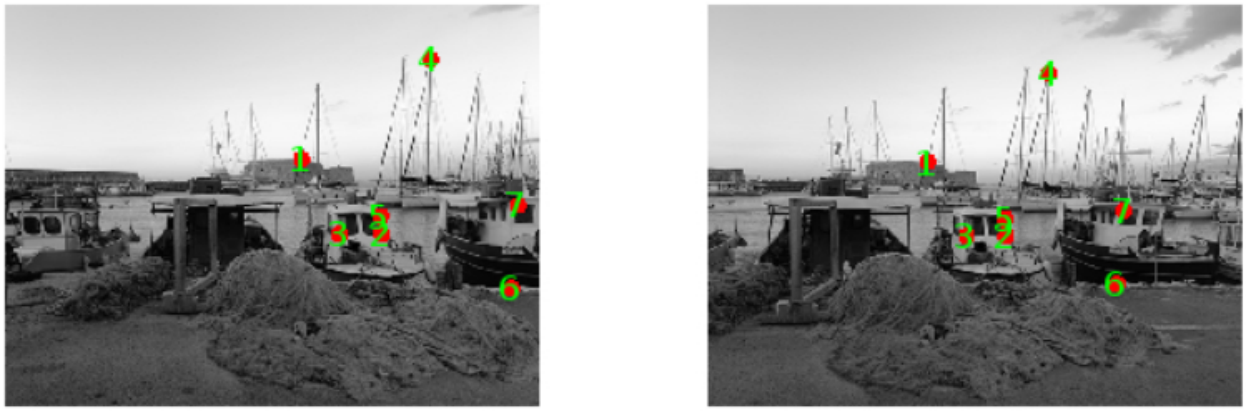

FIGURE 2 – Résultat de l'appariement des points d'intérêt détectés par le détecteur de Harris après application des contraintes de seuil et de symétrie. Nous avons gardé un affichage avec un cercle rouge mais nous ajoutons des numéros pour vérifier les correspondances.

4 Estimation et application de l'homographie

4.1 Présentation de l'homographie

En faisant subir à une caméra une rotation dont l'axe passe par le centre optique, la transformation entre les projections des points de la scène dans les différentes images est une homographie, représentée par la matrice inversible d'ordre 3 :

$$\mathbf{H} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix}.$$

Si nous notons les vecteurs de coordonnées homogènes de deux points homologues $\mathbf{m}_1(x_1, y_1, 1)$ et $\mathbf{m}_2(x_2, y_2, 1)$, nous avons :

$$\lambda \mathbf{m}_2 = \mathbf{H}\mathbf{m}_1, \tag{4}$$

où λ est un facteur non nul. Nous pouvons montrer que parmi les 3 équations vectorielles (4), seulement 2 sont linéairement indépendantes. De plus, la matrice $\mathbf{H}$ étant définie à un facteur près, seuls 8 paramètres sont indépendants. Ainsi, 4 correspondances de points suffisent pour déterminer $\mathbf{H}$. Plus précisément, chaque correspondance de points, numérotée i, donne les 2 équations suivantes :

$$\begin{cases} x_2^i = \dfrac{h_{11}x_1^i + h_{12}y_1^i + h_{13}}{h_{31}x_1^i + h_{32}y_1^i + h_{33}} \\ y_2^i = \dfrac{h_{21}x_1^i + h_{22}y_1^i + h_{23}}{h_{31}x_1^i + h_{32}y_1^i + h_{33}}. \end{cases}$$

que nous pouvons également écrire :

$$\mathbf{a}^i\mathbf{h} = 0 \quad \text{avec}$$

$$\mathbf{a}^i = \begin{pmatrix} x_1^i & y_1^i & 1 & 0 & 0 & 0 & -x_2^i x_1^i & -x_2^i y_1^i & -x_2^i \\ 0 & 0 & 0 & x_1^i & y_1^i & 1 & -y_2^i x_1^i & -y_2^i y_1^i & -y_2^i \end{pmatrix} \tag{5}$$

$$\mathbf{h} = \begin{pmatrix} h_{11} & h_{12} & h_{13} & h_{21} & h_{22} & h_{23} & h_{31} & h_{32} & h_{33} \end{pmatrix}^T$$

Ainsi, avec N correspondances estimées, nous obtenons le système d'équations linéaires homogènes suivant :

$$\mathbf{A}_{2N\times 9}\mathbf{h} \approx 0, \tag{6}$$

où $\mathbf{a_i}$ est une paire de lignes adjacentes de $\mathbf{A}$. Le système n'est plus exactement vérifié car les correspondances sont entachées d'erreurs de mesure. Nous devons alors résoudre le système au sens des moindres carrés en cherchant une solution au problème suivant :

$$\min_{\mathbf{h}} \|\mathbf{A}\mathbf{h}\|^2 \quad \text{s.c.} \quad \|\mathbf{h}\| = 1. \tag{7}$$

La contrainte a pour but d'éviter la solution triviale $\mathbf{h} = 0$. La solution $\hat{\mathbf{h}}$ (estimation de $\mathbf{h}$) est donnée par la décomposition en valeurs singulières, *Singular Value Decomposition*, SVD, ou encore par le vecteur propre de $\mathbf{A}^T\mathbf{A}$ associé à sa plus petite valeur propre (en utilisant la technique des multiplicateurs de Lagrange).

4.2 Estimation

En utilisant les explications du rappel et celles données dans `homographie.m`, complétez l'estimation de l'homographie. Pour tester votre fonction, ≪ dé-commentez ≫ la ligne correspondante dans `TP_mosaique.m`.

Vérification

Avec l'exemple fourni, vous devez obtenir la matrice d'homographie suivante (aux arrondis près) :

```
H =
   -0.0077   -0.0003    0.9559
   -0.0009   -0.0070    0.2936
   -0.0000   -0.0000   -0.0056
```

4.3 Application

Complétez la fonction `appliquerHomographie` (appelée par la fonction `mosaique`) qui permet d'appliquer une homographie à un ensemble de points. Pour tester votre fonction, « dé-commentez » la ligne correspondante faisant appel à la fonction `mosaique` dans `TP_mosaique.m`.

Vérification

L'appel à `mosaique`, via `TP_mosaique.m`, doit générer l'affichage suivant :

```
xy_coinsI2_R1 =
  124.2109   26.4346
  768.3645  -53.3403
  774.6912  507.2647
  108.5470  440.4217
```

5 Construction de la mosaïque

Vérification de la mosaïque

Une fois l'étape d'estimation et d'application de l'homographie validée, il faut vérifier que vous obtenez un résultat similaire à la figure 3.

FIGURE 3 – Construction de la mosaïque en niveaux de gris, à partir de deux images initiales fournies. Nous pouvons observer une démarcation à la jonction entre les deux images.

5.1 Amélioration de la mosaïque

Sur le résultat de la figure 3, nous voyons nettement la démarcation entre les deux images. Cela est dû au fait que nous prenons directement le niveau de gris de l'image droite quand il s'agit d'un point visible dans les deux images. Il existe une autre technique pour calculer le niveau de gris d'un point visible dans les deux images. Elle s'appuie sur le calcul d'une moyenne pondérée, les poids étant fixés

par les distances aux bords des images initiales. Plus précisément, si nous notons N_1^c le nombre de colonnes dans l'image 1, alors les poids p_1 et p_2, respectivement associés aux points de coordonnées (x_1, y_1) et (x_2, y_2) sont donnés par :

$$p_1 = \frac{d_1}{d_1 + d_2} \quad \text{et} \quad p_2 = \frac{d_2}{d_1 + d_2}$$

avec : $d_1 = N_1^c - y_1$, la distance colonne entre le bord droit de l'image I_1 et sa coordonnées colonne y_1 et $d_2 = y_2$, la distance colonne entre le bord gauche de l'image I_2 et sa coordonnées y_2, soit y_2 lui même. Pour comprendre ces explications, vous pouvez vous appuyez sur le schéma de la figure 4.

Effectuez cette modification et comparez à la première mosaïque. **Dans tous vos codes, n'oubliez pas que la convention `matlab` est inversée en terme de colonne et de ligne par rapport à la convention utilisée dans les explications en traitement d'images.**

FIGURE 4 – Construction de la mosaïque. Il n'y a plus de démarcation.

Vérification de la mosaïque améliorée

Une fois cette amélioration réalisée, il faut vérifier que vous obtenez un résultat similaire à la figure 5.

5.2 Mosaïque en couleur

Les images couleur correspondent à trois matrices représentant les canaux rouge, vert et bleu. Si la même transformation s'applique à chacune de ces matrices, il est très simple de réaliser les calculs précédents sur des images couleur.

Écrivez une fonction `mosaiquecoul.m` qui permet de calculer cette mosaïque.

Vérification de la mosaïque en couleur

Pour tester votre fonction « dé-commentez » la ligne correspondante dans `TP_mosaique.m`. Vous devez obtenir un résultat similaire à la figure **??**.(b).

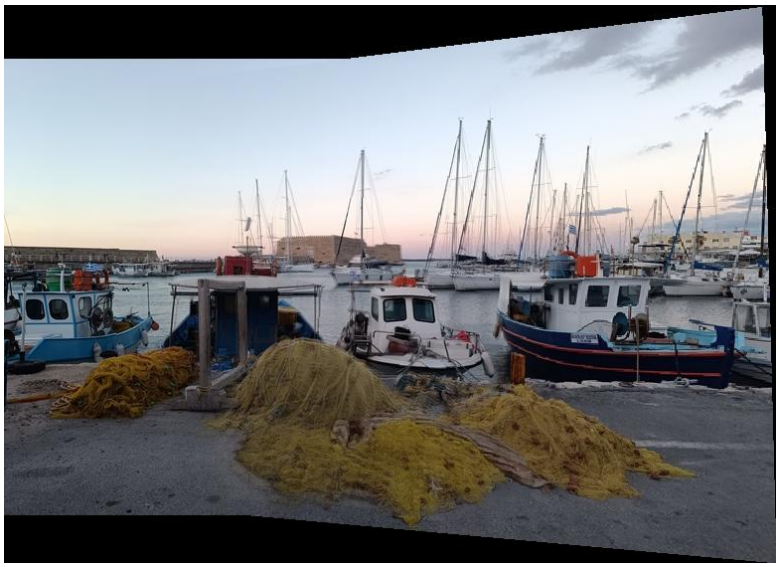

FIGURE 5 – Construction de la mosaïque en niveaux de gris, à partir de deux images initiales fournies. Grâce à l'amélioration réalisée, il n'y a plus de démarcation.

5.3 Mosaïque avec les 3 images

Nous vous proposons à présent de calculer une mosaïque avec les 3 images fournies. Pour cela, vous devez compléter le script en suivant les instructions.