TP3 – Stéréoscopie multi-vues

Formulation générale du MVS

Si l'on désigne par $\mathbf{Q} = [X, Y, Z]^\top$ un point de la surface d'une scène 3D, supposée opaque, et par $\mathbf{q} = [x, y]^\top$ son image formée par une caméra perspective, le lien entre ces deux points s'écrit $\mathbf{q} = \pi(\mathbf{Q})$, où π est une projection centrale. Une projection est en général non inversible, mais en supposant connue la fonction z, dite « fonction de profondeur », telle que $Z = z(x, y)$, il existe bel et bien une bijection entre les points 3D vus par la caméra et leurs images. Nous pouvons dès lors définir la « projection inverse » π_z^{-1} telle que :

$$\mathbf{Q} = \pi_z^{-1}(\mathbf{q}) \tag{1}$$

où l'indice z indique que, sans la connaissance de z, une telle écriture serait ambiguë.

Si nous disposons de $n + 1$ images d'une même scène 3D, nous choisissons une de ces images comme *image de référence*, et définissons la fonction de profondeur z, supposée connue, relativement au repère caméra de cette image. Soit $\mathbf{q} = \pi(\mathbf{Q})$ l'image d'un point $\mathbf{Q}$ de la scène 3D dans l'image de référence. Connaissant les n changements de pose de la caméra correspondant aux n autres images, dites *images témoins*, nous pouvons reprojeter $\mathbf{Q}$ dans ces autres images, ce qui donne les n points image suivants :

$$\mathbf{q}_k = \pi_k(\mathbf{Q}), \qquad k \in \{1, \dots, n\} \tag{2}$$

Ces points sont homologues à $\mathbf{q}$, puisqu'ils correspondent tous au même point $\mathbf{Q}$ de la scène 3D.

La fonction notée $I : \mathbb{R}^2 \to \mathbb{R}$ désigne le niveau de gris de l'image de référence (la fonction `rgb2gray` de Matlab pourra être utilisée pour les images en couleur). Quant au niveau de gris de la $k^{\text{ème}}$ image témoin, il est noté I_k, $k \in \{1, \dots, n\}$. L'*hypothèse lambertienne* $I_k(\mathbf{q}_k) = I(\mathbf{q})$ nous permet d'écrire, en combinant (1) et (2) :

$$I_k \circ \pi_k \circ \pi_z^{-1}(\mathbf{q}) = I(\mathbf{q}), \quad k \in \{1, \dots, n\} \tag{3}$$

Cette égalité devant être vérifiée en tout point $\mathbf{q} = [x, y]^\top$ de l'image de référence, nous pouvons la réécrire :

$$I_k \circ \pi_k \circ \pi_z^{-1}(x, y) = I(x, y), \quad k \in \{1, \dots, n\}, \quad (x, y) \in \Omega \tag{4}$$

où $\Omega \subset \mathbb{R}^2$ désigne l'ensemble des points de l'image de référence qui sont visibles dans toutes les autres images. En pratique, l'égalité (4) n'est jamais parfaitement vérifiée. Dans une démarche de reconstruction 3D où l'inconnue est z, nous pouvons la reformuler sous la forme d'un problème variationnel :

$$\min_{z\colon \Omega \to \mathbb{R}} \sum_{k=1}^{n} \| \, I_k \circ \pi_k \circ \pi_z^{-1} - I \, \|^2_{\ell^2(\Omega)} \tag{5}$$

où $\| \cdot \|^2_{\ell^2(\Omega)}$ désigne l'intégrale sur Ω de la norme L^2. Le problème (5) constitue le modèle le plus élémentaire de *stéréoscopie multi-vues* (en anglais MVS, pour *multi-view stereo*). Notons que, en comparaison des problèmes d'optimisation déjà rencontrés, l'inconnue du problème (5) est *continue*, et non discrète. Nous pouvons donc légitimement espérer que cette technique fournisse des reconstructions 3D *denses*, contrairement au SfM. En contrepartie, la résolution risque d'être plus délicate. Commençons par approcher l'intégrale de la norme L^2 dans (5) par une somme discrète, ce qui est d'autant plus justifié que les images numériques sont constituées de pixels $\mathbf{p}$:

$$\min_{z\colon \Omega \to \mathbb{R}} \sum_{k=1}^{n} \sum_{\mathbf{p} \in \Omega} \left\| \, \mathbf{I}_k \circ \pi_k \circ \pi_z^{-1}(\mathbf{p}) - \mathbf{I}(\mathbf{p}) \right\|^2 \tag{6}$$

où $\mathbf{I}$ désigne la fonction vectorielle obtenue en vectorisant les niveaux de gris des pixels contenus dans un voisinage de taille $t \times t$ centré en $\mathbf{p}$, avec $t \in \mathbb{N}^*$, et de même pour les fonctions $\mathbf{I}_k$, $k \in \{1, \dots, n\}$.

Résolution du MVS

Le problème (6) peut être résolu pixel par pixel, ce qui revient à résoudre, pour chaque pixel $\mathbf{p} \in \Omega$:

$$\min_{z_{i,j} \in \mathbb{R}} \sum_{k=1}^{n} \left\| \mathbf{I}_k \circ \pi_k \circ \pi_{z_{i,j}}^{-1}(\mathbf{p}) - \mathbf{I}(\mathbf{p}) \right\|^2 \tag{7}$$

où $z_{i,j}$ désigne la profondeur $z(\mathbf{p})$ de $\mathbf{p}$, repéré par son numéro de ligne i et son numéro de colonne j. Attention : ces indices diffèrent des coordonnées $[u, v]^\top$ de $\mathbf{p}$ exprimées dans le repère pixels !

Vu que l'inconnue $z_{i,j}$ du problème (7) n'apparaît pas de manière directe, mais au travers de la combinaison de fonctions $\mathbf{I}_k \circ \pi_k \circ \pi_{z_{i,j}}^{-1}$, il semble difficile d'utiliser les outils de l'optimisation différentiable tels que le gradient ou la hessienne. En outre, il est probable que le point $\pi_k \circ \pi_{z_{i,j}}^{-1}(\mathbf{p})$ ne se situe pas exactement au centre d'un pixel de la $k^{\text{ème}}$ image témoin. L'expression $\mathbf{I}_k \circ \pi_k \circ \pi_{z_{i,j}}^{-1}(\mathbf{p})$ doit donc être calculée par interpolation. Enfin, il s'avère que l'argument du problème (7) n'évolue pas du tout de façon régulière en fonction de $z_{i,j} \in \mathbb{R}$. Pour toutes ces raisons, nous recherchons la solution en testant N valeurs de $z_{i,j}$ notées v_1, …, v_N, ce qui revient à résoudre ce problème d'optimisation par une recherche exhaustive de la solution (recherche « en force brute ») :

$$\min_{z_{i,j} \in \{v_1, \ldots, v_N\}} \rho\left(\left\| \mathbf{I}_1 \circ \pi_1 \circ \pi_{z_{i,j}}^{-1}(\mathbf{p}) - \mathbf{I}(\mathbf{p}) \right\|, \ldots, \left\| \mathbf{I}_n \circ \pi_n \circ \pi_{z_{i,j}}^{-1}(\mathbf{p}) - \mathbf{I}(\mathbf{p}) \right\| \right) \tag{8}$$

Le problème (8) est équivalent à (7) si $\rho(a_1, \ldots, a_n)$ est égal à la somme des carrés de ses arguments. Or, il est connu que les moindres carrés ne constituent pas un estimateur robuste. Il est préférable d'utiliser, par exemple :

$$\rho_{\text{robuste}}(a_1, \ldots, a_n) = 1 - \exp\left(-\frac{\sum_{k=1}^{n} a_k^2}{0,2^2} \right) \tag{9}$$

Exercice 1 : MVS sur images de synthèse

Le fichier `lapin.mat` contient $n + 1$ images de synthèse I_0, …, I_n d'un lapin et la matrice de calibrage $\mathbf{K}$. Les poses de la caméra, exprimées dans un « repère monde », vous sont également fournies.

Écrivez la fonction `MVS`, appelée par le script `exercice_1`, permettant de calculer l'argument du problème (8), en calculant $\mathbf{I}_k \circ \pi_k \circ \pi_z^{-1}(\mathbf{p})$ par interpolation, par exemple au plus proche voisin (fonction `round` de Matlab).

Remarques

- Dans ce premier exercice, la comparaison des niveaux de gris est effectuée pixel à pixel, c'est-à-dire que les fonctions $\mathbf{I}$ et $\mathbf{I}_k$, $k \in \{1, \ldots, n\}$, sont censées être à valeurs dans $\mathbb{R}$.

- Après *déprojection* d'un pixel de l'image de référence, et avant *reprojection* dans chacune des images témoins, il est nécessaire de calculer les coordonnées du point 3D dans le « repère monde ». Il est donc nécessaire d'effectuer deux changements de repère successifs : $\mathcal{R}_{\text{réf}} \to \mathcal{R}_{\text{monde}}$, puis $\mathcal{R}_{\text{monde}} \to \mathcal{R}_k$.

Exercice 2 : utilisation de fenêtres de comparaison de taille 3×3

La comparaison des niveaux de gris pixel à pixel ne peut fournir des résultats très précis, surtout pour les images de synthèse d'une surface lisse sans texture, ce qui est le cas des données du fichier `lapin.mat`. La reconstruction 3D sera forcément meilleure en comparant les niveaux de gris des pixels contenus dans une fenêtre centrée de taille 3×3. Dorénavant, les fonctions $\mathbf{I}$ et $\mathbf{I}_k$, $k \in \{1, \ldots, n\}$, du problème (8) devront donc être à valeurs dans $\mathbb{R}^9$. Pour ce faire, une astuce consiste à concaténer, à chacune des $n + 1$ images du lapin, huit images obtenues par décalages dans les huit directions cardinales N/E/S/O et intercardinales NO/NE/SE/SO. Cette astuce vous sera détaillée pendant la séance de TP.

Écrivez les fonctions `pretraitement` et `MVS_2`, appelées par le script `exercice_2`, de manière à mettre en œuvre cette variante plus performante de MVS.

Exercice 3 : MVS sur images réelles

Le fichier de données `fontaine.mat` contient $n > 2$ images réelles en couleur d'une fontaine, les masques associés à ces images, qui « détourent » la fontaine (le sol et les éléments du fond ont été escamotés), la matrice de calibrage de la caméra et les n poses de la caméra, relativement à un repère commun appelé « repère monde ».

Faites une copie du script `exercice_2`, de nom `exercice_3`, que vous modifierez de manière à effectuer la reconstruction 3D dense de la fontaine par MVS, à partir des données contenues dans ce fichier.

Remarques

- Dans la phase de mise au point, il pourra être utile de redimensionner les images à l'aide de la fonction `imresize` de Matlab, afin d'accélérer les calculs.
- Vous pourriez être tentés d'utiliser les codes des deux premières séances de TP pour estimer les différentes poses de la caméra. Néanmoins, les estimations ainsi obtenues ne seraient pas suffisamment précises pour obtenir une reconstruction 3D dense de qualité « acceptable ».
- Il serait appréciable de pouvoir déterminer automatiquement la plage de valeurs de la profondeur, mais dans un premier temps, cees valeurs vous sont fournies : si l'image de la fontaine choisie comme image de référence est l'image numéro 4, ces valeurs sont : `valeurs_z = 7:0.01:10;`
- Les images doivent être converties en niveaux de gris pour le calcul de l'erreur de reprojection, mais il est nécessaire de garder une version en couleur de l'image de référence pour l'affichage du résultat.
- L'affichage du nuage de points 3D colorés doit être effectué à l'aide de la fonction `affichage_nuage`.