Projets du module Vision 3D

Théodore Papadopoulo

26 février 2000

Les règles du jeu :

Ces projets sont a faire seuls. Je donnerai une prime à l'originalité, donc ne me faites pas tous la même chose. Le langage utilisé m'importe peu, mais consultez moi si vous voulez utiliser autre chose que C, C++, Matlab, Maple ou Java. Pour chacun des trois sujets, j'ai indiqué une liste d'extensions possibles. J'aimerais bien que chacun d'entre vous s'attaque à au moins une de ces questions subsidiaires. J'attends au final un court rapport comportant un listing du programme ainsi que quelques résultats commentés (les difficultés rencontrées, la qualité du résultat, propositions d'améliorations éventuelles, etc). Pas plus que 4 ou 5 pages (hors listing). Pensez à écrire du code lisible et commenté. N'hésitez pas à expérimenter des choses, ces projets sont faits pour cela. Le source du programme devra aussi m'être fourni par FTP.

1 Stéréo-Vision par corrélation

Il s'agit d'écrire un programme de reconstruction par stéréo-vision se basant directement sur les images d'intensité. Le principe de la méthode est simple. On suppose qu'on dispose d'un système calibré de 2 caméras qui observe une scène. Pour des raisons de simplicité, on supposera d'abord que les images de départ auront été rectifiées au préalable et donc que les droites épipolaires sont des horizontales: la droite de l'image 2 associée au point de l'image 1 (x_1, y_1) est la droite $y_2 = y_1$ et réciproquement la droite épipolaire de l'image 1 associée au point de l'image 2 (x_2, y_2) est la droite $y_1 = y_2$.

Le principe de la corrélation est de considérer autour d'un point de l'image 1 (x_1, y_1) une fenêtre (qu'on supposera carrée) de taille w. Pour chaque point sur l'épipolaire correspondante dans l'image 2 $(x_2 + d, y_2)$, on va également considérer un fenêtre de taille w et calculer un critère de ressemblance de ces deux fenêtres. Trouver le meilleur appariement pour le point (x_1, y_1) consiste à trouver la valeur de d qui optimise ce critère. d est appelée la disparité associée au point (x_1, y_1). L'image qui associe au point (x_1, y_1) la valeur d est appelée carte de disparités entre les images 1 et 2. Noter que cette carte est relative à l'image 1 qui est utilisée comme image de référence.

Le but du projet est d'obtenir cette carte de disparité. Il est recommandé de soigner le codage afin que le programme obtenu puisse être facilement modifié (en particulier pour changer le critère de corrélation). Les temps d'éxécution d'une stéreo par corrélation peuvent être assez long, penser à travailler avec des imagettes 128x128 par exemple quand vous ferez des tests. On a souvent une idée des disparités que l'on cherche. Faire en sorte de pouvoir spécifier l'intervalle des disparités recherchées afin de réduire les temps de calcul.

Essayez votre algorithme sur les données en `Data/Stereo`. Il y a dans ce répertoire plusieurs images:

- Une paire d'images d'un visage `herveleft.gif` et `herveright.gif`.
- Deux stéréogrammes différents (images à corréler avec elle-même) `3d.pgm` et `3d_3.pgm`. Un exemple de résultat est donné en `3d_3_result.gif`.

Essayer différents critères (attention, suivant le critère, l'optimum correspond soit à une maximisation soit à une minimisation du critère) :

- • $SSD(x,y) = \sum_i \sum_j (I_1(x_i,y_j) - I_2(x_i,y_j))^2$
- • $ZSSD(x,y) = \sum_i \sum_j ((I_1(x_i,y_j) - \overline{I_1(x,y)}) - (I_2(x_i,y_j) - \overline{I_2(x+d,y)}))^2$
- • $ZNSSD(x,y) = \frac{\sum_i \sum_j ((I_1(x_i,y_j) - \overline{I_1(x,y)}) - (I_2(x_i,y_j) - \overline{I_2(x+d,y)}))^2}{\sqrt{\sum_i \sum_j (I_1(x_i,y_j) - \overline{I_1(x,y)})^2 \sum_i \sum_j (I_2(x_i,y_j) - \overline{I_2(x+d,y)})^2}}$
- • $CC(x,y) = \sum_i \sum_j (I_1(x_i,y_j))(I_2(x_i,y_j))$
- • $ZNCC(x,y) = \frac{\sum_i \sum_j (I_1(x_i,y_j) - \overline{I_1(x,y)})(I_2(x_i,y_j) - \overline{I_2(x+d,y)})}{\sqrt{\sum_i \sum_j (I_1(x_i,y_j) - \overline{I_1(x,y)})^2 \sum_i \sum_j (I_2(x_i,y_j) - \overline{I_2(x+d,y)})^2}}$

les critères SSD, $ZSSD$, $ZNSSD$, d'une part et CC et $ZNCC$ d'autre part ne diffèrent que par des normalisations locales sur les images. Les sommations sur les indices i et j correspondent à la sommation sur la fenêtre de taille w. Par exemple, pour des fenêtres de taille 3 centrées au point (x,y), les indices i et j sont compris entre -1 et $+1$. On a alors $x_i = x + i$ et $y_i = y + i$ dans l'image 1 et $x_i = x + d + i$ et $y_i = y + i$ dans l'image 2. $\overline{I}(x,y)$ désigne la moyenne locale de l'image sur la fenêtre considérée centrée en (x,y).

Dans un deuxième temps, on suppose que les images ne sont plus rectifiées. Ecrire un programme de rectification (Voir les chapitres de thèse de Frédéric Devernay, par exemple). Utilisez ce programme et votre programme de stéréo par corrélation pour traiter les images non rectifiées.

Commentez les résultats que vous obtenez.

1.1 Questions subsidiaires

Ces suggestions sont vaguement dans un ordre de complexité croissant. Ces suggestions sont totalement indépendantes les unes des autres.

- Prévoir une valeur minimale (dans un sens à définir suivant le critère utilisé) pour laquelle on considérera que les points sont en correspondance. Si cette "qualité" minimale n'est pas atteinte, alors prévoir une valeur spéciale de la disparité qui signifiera "Pas de correspondant".

- Optimiser les calculs pour éliminer les calculs redondants. Regarder les calculs effectués sur deux fenêtres voisines et en déduire que les calculs peuvent être faits de manière incrémentale.

- Améliorer la localisation de la correspondance : Pour cela on supposera que le critère est localement un polynôme de degré 2 autour de l'optimum recherché. Calculer ce polynôme de degré 2 et extraire son optimum. La disparité obtenue sera alors une valeur flottante donc attention à la visualisation de la carte de disparité.

- Implémenter la corrélation aller-retour: tel que décrit ci-dessus, l'algorithme privilégie l'image 1 en ce sens que pour chaque point de cette image on calcule un meilleur correspondant dans l'image 2, mais rien ne garantit que pour ce point de l'image 2, il n'existe pas un point de l'image 1 qui lui corresponde mieux. Dans la corrélation aller-retour, on considère que des points sont en correspondance que si il se "choisissent" mutuellement.

- Implémenter la corrélation en utilisant une pyramide d'image, la carte de disparité obtenue à une résolution grossière servant à initialiser l'intervalle de recherche à la résolution suivante (plus dur).

D'autres extensions/améliorations sont possibles, si vous avez des envies n'hésitez pas à m'en parler...

2 Calcul de mosaïques

On suppose que l'on a un ensemble d'images prise de telle sorte qu'il est possible de créer une mosaïque. Créer un programme qui va calculer les homographies relatives entre toutes ces images et qui créera une image mosaïque (choisir une image de référence et "replacer" toutes les autres images par rapport à cette référence). Implémenter l'algorithme aux moindres carrés avec minimisation non-linéaire des résidus. Attention, je ne m'attend pas a ce que vous programmiez une minimisation non-linéaire, il en existe plusieurs dans la nature Matlab, minpack, numerical recipies, ...

Tester les algorithmes sur les images se trouvant en `Data/Mosaics`. Il y a dans ce répertoire plusieurs images:

- Cinq images d'une église (i.xx) dans le catalogue `Eglise`. Pour cette deuxième série d'images des appariements de points sont donnés dans le fichier `i.matchs`. La syntaxe de ce fichier est simple. Si l'on néglige l'en-tête (les 10 premières lignes), il y a un appariement par ligne. Sur chaque ligne, on trouve une liste de coordonnées préfixées par un index d'image (la correspondance entre index d'image et les images est donnée dans l'en-tête). Ainsi, `1 183.415662629 422.47 0 687.11746988 428.485454545` est interprété comme: le point de coordonnées $(687.11746988, 428.485454545)$ de l'image 0 (`i.25`) correspond au point de coordonnées $(183.415662629, 422.47)$ de l'image 1 (`i.26`).

- Cinq images d'une villa (avec piscine pour rêver de vacances!) dans le catalogue `Piscine`. Il faudra cliquer les points. Un exemple de mosaïque crée avec ces images est montré en `mosaic.ppm`.

Rajoutez des appariements erronés. Comment se dégradent les résultats ?

2.1 Questions subsidiaires

Ces suggestions sont totalement indépendantes les unes des autres et de complexités équivalentes (à mon avis).

- En supposant que les paramètres intrinsèques sont constants, calculer ceux-ci à partir des homographies à partir de la méthode vu en cours (cas particulier de l'auto-calibration pour le cas de la rotation pure).

- Implémenter un algorithme résistant aux outliers en s'inspirant d'une des techniques vues pour le calcul de la matrice fondamentale.

- Créer un mini-film en se promenant dans la mosaïque.

- Améliorer la qualité du rendu aux jointures entre les images.

- Rajouter un module d'appariement automatique en s'inspirant des techniques de corrélation du premier mini-projet (plus dur).

Dans ce cas aussi, d'autres extensions/améliorations sont possibles, si vous avez des envies n'hésitez pas à m'en parler... Je suis prêt à considérer d'autres choses si le coeur vous en dit.

3 Auto-calibration

Le but de ce projet est de créer un programme permettant de faire de l'autocalibration. Les différentes étapes de ce programme sont:

- Calcul de la matrice fondamentale à partir d'ensembles de points appariés.
- Ecriture puis résolution des équations de Kruppa pour obtenir la matrice des paramètres intrinsèques (on suppose ceux-ci constants).
- Calcul des paramètres extrinsèques des caméras.

On suppose que les appariements de points sont donnés manuellement (je vous laisse cliquer les points). On pourra donc se contenter d'une estimation de la matrice fondamentale simple (linéaire + raffinement non-linéaire) sans avoir à utiliser les techniques robustes a la least-median squares..

Pour faciliter la résolution du système d'équations, on supposera que les axes des caméras sont orthogonaux et que le point principal est situé au centre de l'image. La résolution des équations se fera alors aux moindres carrés. Dans un deuxième temps, on pourra faire varier tous les paramètres.

3.1 Questions subsidiaires

Ces suggestions sont totalement indépendantes les unes des autres et de complexités équivalentes (à mon avis).

- Rajouter un module d'appariement automatique en s'inspirant des techniques de corrélation du premier projet (corrélation). On supposera d'abord que les points sont toujours "cliqués" et donc que pour chaque point de la première image il existe un point de la deuxième qui lui correspond. Dans un deuxième temps, il est possible de relaxer cette contrainte.
- Implémenter un algorithme résistant aux outliers en s'inspirant d'une des techniques vues pour le calcul de la matrice fondamentale (introduisez manuellement quelques appariements erronés et vérifiez que ceux-ci sont correctement détectés).
- A partir des données images, on peut définir des polygônes correspondants grossièrement aux plans de la scène. La reconstruction 3D des points sommets des polygônes permet alors d'avoir une reconstruction 3D de la scène. Faire une telle reconstruction et visualiser d'autres points de vues. Si vous avez du courage, vous pouvez même générer du VRML et texturer la reconstruction 3D en prenant l'information dans les images (cette deuxième partie demande plus de travail qu'une simple reconstruction).

Dans ce cas aussi, d'autres extensions/améliorations sont possibles, si vous avez des envies n'hésitez pas à m'en parler… Je suis prêt à considérer d'autres choses si le coeur vous en dit.