Rapport de projet Structure From Motion Arnaud DALIE SY32 P18

Introduction : Rappel du sujet

Pour le semestre de P18 et pour illustrer la dernière partie de SY32, il nous a été demander de travailler sur le projet Strucutre-From-Motion. Ce projet de reconstruction 3D consiste donc a mettre en place toutes les étapes permettant d'estimer la position de points en 3D à partir d'une paire d'images. La démarche qui nous a été conseillé pour ce travail a été d'appliquer dans un premier temps le tutoriel disponibles dans la documentation matlab afin d'aboutir à un algorithme fonctionnel, puis de remplacer progressivement ces fonctions par d'autres que nous aurions développé nous-même. Par manque de temps et de disponibilité des salles informatiques, je ne serais pas en mesure de fournir des captures d'écrans de l'output de mon code dans ce rapport. Dans ce document, nous présenterons brièvement le data set puis quels sont les fichiers du projet et comment ils s'articulent entre eux. Par la suite nous nous pencherrons sur le code en lui-même. Nous passerons moins de temps sur les fichiers vus en cours ou en TD (estimation de la matrice fondamentale), ou encore ceux décrits dans le sujet (detecteur de Harris).

Presentation du dataset

Pour commencer, nous allons aborder le sujet du dataset utilisé qui a eu une influence sur la manière d'arborer le projet. Le data set choisi est celui du projet Karlsruhe (2010_03_04_drive_0042.zip). Ce data set consiste en une séquence d'images stereo capturées par des caméras placées sur des voitures en mouvement. Il y avait deux possibilités pour le projet, soit utiliser les images d'une seule caméra et d'utiliser des paires d'images successives avec un décalage Δ_t ; soit en utilisant des caméras en stéréo. C'est la deuxième solution qui a été choisi, car cela nous permet d'avoir des rotations et translation entre les caméras constantes et que nous n'avions pas les données GPS pour chaque image du dataset. Le dataset nous permet de connaître les matrices de projection des deux caméras ainsi que la rotation et la translation entre les deux caméras. Nous reviendrons sur ce qu'a impliqué la connaissance de cet élément.

Structure du projet

Le projet rendu comprend 8 fichiers matlab. Nous allons les décrires afin d'expliquer comment ils s'articulent entre eux.

En premier lieu, le fichier tuto.m est le fichier issu du tuto structure from motion de matlab mais adapté au dataset que l'on a utilisé. Les fichiers normalize.m et FundamentalMatrix.m sont des fichiers issus des TD. En effet, le premier nous a été envoyé par E.Cappelier et le second est le fruit de mon travail pendant une séance de TD. Cependant puisque les matrices de projections sont données dans le dataset, nous n'avons pas besoin de l'utiliser mais il me sembler pertinent de l'ajouter au projet rendu. Pour finir, le fichier main.m est le fichier central qui appelle successivement les fonctions loadImages.m qui charge les images, HarrisDetect.m qui fait la detection des points d'intêret dans l'image, Correlation.m qui associe les poins détectés entre eux et en fin triangulatee.m qui resitue les points dans l'espace 3D à partir de la matrice des matrices de projection. Le reste du fichier main est directement issu de la documentation matlab et permet l'affichage des points dans un repère 3D.

Description des fonctions principales

Cette partie va aborder les quatres fonctions principales du projet qui correspondent aux étapes de détections des points d'intêret, du matching de ces points puis de la triangulation de ces points à partir des matrices de projection.

Detecteur de Harris

La première étape de l'algorithme est la detection des points d'intêret. Pour cela nous avons mis en place un detecteur de Harris en suivant les directives du sujet. On commence par détecter les contours de l'image en y appliquant un masque de Prewitt par convolution. On extrait ensuite pour chaque pixel de l'image (noir et blanc) d'origine une fenêtre centré sur le pixel. Cette fenêtre sera transormé par un filtre gaussien et servira à calculer la matrice d'autocorrélation M pour chaque pixel. Avec tout cela, on peut calculer le $R(x, y)$ de l'image pour chaque pixel de coordonnées (x, y). On extrait les coordonnées des points ayant les valeurs de $R(x, y)$ les plus hautes.

Au niveau des paramètres influant sur la fonction, on a le masque utilisé qui peut faire varier les points détectés, la taille de la fenêtre utilisé pour calculer l'auto-correlation, le k de la formule servant à determiner R qui est usuellement compris entre 0.04 et 0.06 et enfin le nombre de point renvoyé.

Les resultats renvoyés par ce detecteur sont corrects mais trop regroupés. En effet, la version de matlab ne va pas détecter des groupes de points contrairement à celui-ci. Le résultat est la présence de nombreux points sur les branches d'arbres et sur les phares des voitures. Cela est probablement dû à la taille de la fenêtre qui est trop petite ou encore au flou appliqué qui n'est pas assez puissant.

Correlation des points

A ce stade de l'algorithme, on a un ensemble de points d'intêret. La fonction de correlation est découpée en deux parties et va se charger de former des paires de points cohérentes entre les deux images.

La première partie de la fonction est le calcul, pour chaque point, de la 'Sum of square difference' (SSD) avec l'ensemble des autres points d'intêret. C'est à dire que pour chaque point, on définit une fenêtre centrée sur ce point et on calcul la SSD avec les sous-fenêtres de tous les autres points de la seconde image. On a après calcul une matrice donnant les SSD pour tous les appariements possibles (ce qui peut être très volumineux selon le nombre de points étudiés). Dans un second temps on va extraire les indices de cette matrice triée par SSD croissante. On a donc un tableau d'indices trié. On va parcourrir se tableau pour faire en sorte d'éliminer les paires ayant des points en commun. C'est à dire que chaque point de l'image 1 ne sera relié qu'à un seul point de l'image 2. On a notre matching qui est fait.

La seconde partie de l'algorithme consiste en une élimination d'outlier. Là où matlab utilise un RANSAC, on utilise ici une programmation "naïve" reprenant plus ou moins la même idée. Le RANSAC cherche à estimer successivement un modèle et éliminer les outlier puis recalculer un autre modèle pour ré-éliminner des outlier et ainsi de suite. Ici, on se contente simplement de calculer pour chaque paire de point un vecteur en faisant la difference des coordonnées. Ensuite on calcule une norme médiane et un vecteur normalisé moyen.

De manière très empirique (trop), on estime qu'une paire de points forment un outlier si la distance entre ces deux points est suppérieure à 5 fois la norme médiane ou si le cosinus de l'angle formé avec le vecteur moyen est inferieur à 0.9.

On a donc au final un ensemble de paire que l'on estime sûre.

Triangulation

La triangulation consiste à calculer la position 3D du point grâce au coordonnées des points dans les plans de projection de caméras dont on connaît la position dans le repère monde. Pour ce faire, on a utilisé la triangulation par l'algorithme DLT qui consiste en somme à trouver le noyeau d'une matrice 4x4 composée de combinaisons entre les coordonées du point dans les deux projection et les matrices de projections associées.

Le fichier triangulatee.m ne fait que mettre en place ce système linéaire et le résoudre par SVD. Ensuite, on supprime le facteur d'échelle en passant la composante des coordonnées homogènes à 1. On a alors, en repassant en coordonées cartésiennes, l'emplacement de notre point dans le repère monde.

Fundamental Matrix

J'ai choisi de parler de cette fonction de manière à part car elle n'est pas utilisée. En tant normal, on aurait utilisé cette fonction pour estimer la matrice F et ensuite retrouver la matrice P. Ici, nous avons déjà P donnée par le dataset. La fonction est une implémentation de l'algorithme des 8 points. On selectionne 8 paires au hasard dans les points en corrélation. Vu que cette estimation repose sur le hasard, il serait peut-être judicieux d'estimer plusieurs fois la matrice fondamentale et de calculer une moyenne au facteur d'echelle près.

Conclusion

Pour conclure, ce projet a rempli son objectif d'aborder beaucoup de concepts liés à la vision par ordinateur. Bien que l'on puisse se satisfaire du fait que l'algorithme fonctionne sans fonctions natives à matlab (on a des points cohérents à des deistances cohérentes). Cependant, il reste un nombre de points assez important sur lesquels il faudrait travailler. Il faudrait passer certains calculs en matriciel afin de permettre à matlab d'optimiser le temps de calcul. Par manque de temps, je n'ai pas pu travailler sur les différents paramètres comme les masques utilisées pour le detecteur de Harris ou la taille des fenêtres. Il aurait été également préférable d'implémenter un vrai RANSAC pour les outliers. Enfin, l'amélioration majeure qu'il reste à faire c'est de pouvoir gérer toutes les étapes du structure From Motion comme l'auto-calibrage ou le bundle adjustment.