Programmation graphique et applications industrielles

R. Raffin

v. 2019

Sommaire intermédiaire

planning

6 séances cours/machines

- 25/09 CM 9h30-12h / TDM 14h-16h
- 01/10 CM 9h30-12h / TDM 13h-15h
- 22/10 CM 9h-12h / TDM 14h-17h
- 05/11 CM 9h-12h / TDM 13h-16h
- 19/11 CM 9h-12h / TDM 13h-16h
- 26/11 CM 9h-12h / TDM 13h-16h

- <évaluation> 04/12 CM 9-12h / TDM 13h-16h </évaluation>

plan

1. numérisation
2. régularisation / analyse de surface
3. sortie visualisation OpenGL/visualisation progressive
4. sortie visualisation WebGL
5. sélection de zones/indexation
6. assemblages d'objets, perçage
7. préparation à l'impression 3D
8. animation, dynamique (collisions)
9. intégration dans un moteur de jeu
10. (comment/pourquoi paralléliser ?)
11. (comment/pourquoi utiliser le GPU ?)

Le module est un grand projet. Chaque séance revient sur les avancées précédentes, définit le travail suivant à effectuer et permet de mettre en place les méthodes, selon les besoins.
Le lien se fait sur Ametice, par groupe. Il faut remplir un wiki au fur et à mesure :

- qu'est-ce qui doit être fait ? qui à fait quoi ? quelles recherches ont été faites ? comment le besoin a été résolu (logiciels, développement) ? quels algorithmes (et pourquoi) ? et l'analyse contradictoire des résultats.
- ajouter images, sources .zip, objets initiaux et résultats
- on fait le point à chaque session de CM/TD

Étape 1 : choix de la numérisation

(objet et méthode)
Pour numériser un objet on peut utiliser plusieurs méthodes, avec chacune leurs propriétés, par exemple :

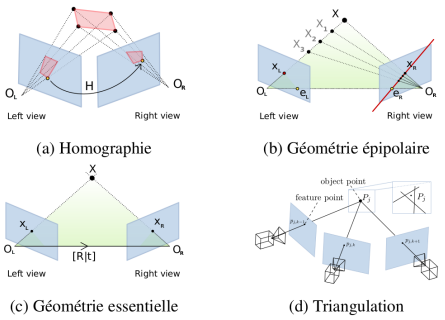
- photogrammétrie
- laser
- temps de vol

Sommaire intermédiaire

Quelques informations sur la photogrammétrie

La photogrammétrie repose sur l'utilisation d'images différentes (de points de vue) d'un même objet. Grâce à plusieurs méthodes (SIFT, traitement des contours) on essaie de caractériser des (groupes de) pixels identiques.
Cela permet d'apparier des éléments des images entre elles, de déduire (éventuellement) des orientations et les paramètres de prise de vue (ouverture, focale), et donc des configuration des appareils de prise de vue.
On utilise ensuite la géométrie épipolaire pour, pour chaque pixel de l'image d'une pose, avoir un nuage de profondeurs. On fusionne ensuite ces nuages pour définir un nuage de points de la scène photographiée.

Des méthodes de photogrammétrie



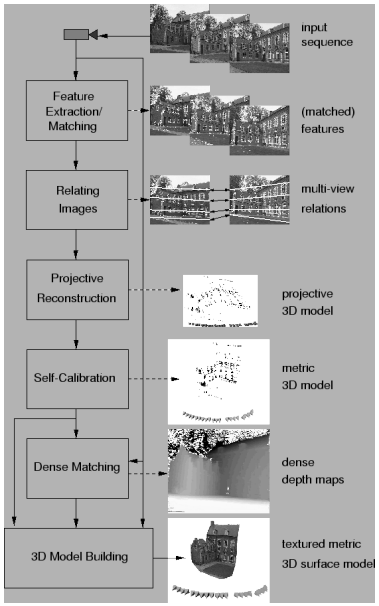
(a) Homographie (b) Géométrie épipolaire

(c) Géométrie essentielle (d) Triangulation

FIGURE 1 – Estimation de modèle.

Pierre Moulon, Pascal Monasse, Renaud Marlet. La bibliothèque openMVG : open source Multiple View Geometry. Orasis, Congrès des jeunes chercheurs en vision par ordinateur, Jun 2013, Cluny, France. <hal-00829332>

Pipeline

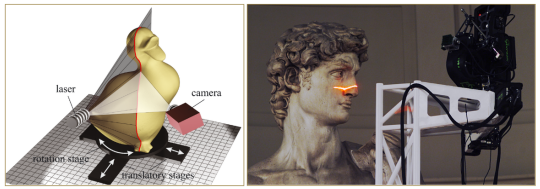


Quelques liens

- `http://fablabo.net/wiki/Scanner_DIY`
- SIGGRAPH Course 2009 Build Your Own 3D Scanner : Optical Triangulation for Beginners `http://mesh.brown.edu/byo3d/index.html`

Scanner Laser

Ici le principe est "d'éclairer" la surface de l'objet à l'aide d'une lumière structurée (type laser ou de forte intensité sur une très fine ouverture). Une (ou plusieurs) caméra(s) calibrée(s) (par rapport à la source) observe(nt) les impacts sur la surface, le modèle de caméra permet d'obtenir les profondeurs des pixels vus.



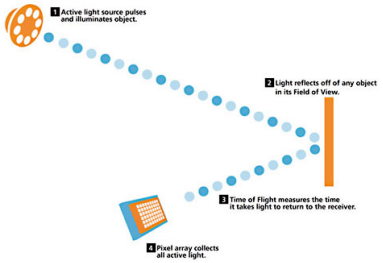
En deux passes on peut obtenir 3D+couleur.

Quelques exemples de matériels :

Les logiciels fournis permettent le recalage à la volée des points obtenus (centrale inertielle parfois embarquée pour l'orientation).
Par exemple : 3DSystem, Geomagic `https://fr.3dsystems.com` ou TechnoDigit 3D Reshaper (`https://www.3dreshaper.com/`).

Scanner à temps de vol

Les scanners à temps de vol ("time of flight") permettent également de calculer une distance à l'objet, en comptant le temps de retour d'une onde pulsée par le scanner. Le décompte du temps doit être très précis (env. 3 nanosecondes pour 1 mètre à 300 000 km/s). La surface de réception doit être assez large pour "attraper" l'onde de retour. Ce système a été popularisé par les Kinect de Microsoft XBox.



Quelques matériels

Figure – Microsoft Kinect, HP Sprout...

Quelques liens

- https://fr.3dsystems.com/3d-scanners/sense-scanner
- https://www.instructables.com/id/EASY-Kinect-3D-Scanner/
- https://realsense.intel.com/

Quelles contraintes ?

Outre le fait d'avoir accès physiquement à l'objet à numériser, la numérisation pose également les contraintes d'éclairage, de réflexion spéculaire du matériau, de sa granularité, de sa texture.
Évidemment, il faut également avoir accès à toute la pièce (dessus, dessous, trous borgnes ou débouchants, "ombres" portées).

La fusion de différents nuages de points n'est pas triviale (recherche des points les plus proches, et des transformations minimales - 3 rotations, une translation, recherche d'amers comparables).
Il faut passer du repère local de chaque nuage à un repère global. C'est généralement une approche au moindres carrés qui est utilisée (cf. "Iterative Closest Point" https://en.wikipedia.org/wiki/Iterative_closest_point).

Sommaire intermédiaire

- exemple de numérisation (une bûche avec IGN MicMac)https://www.youtube.com/watch?v=ikSMTmz3MOA
- pas à pas (snap Ubuntu Mardy) https://www.youtube.com/watch?v=ELHOjC_V-FE
- utilisation de visualSFM http://combiencaporte.blogspot.com/2012/07/la-photogrammetrie-visualsfm-et-meshlab.html

Initialisation de VSfM

Réf. https://www.youtube.com/watch?v=ELHOjC_V-FE

Installation Ubuntu et Debian : snap install...mardy

colmap-mardy, mve-mardy, line3dpp-mardy, bundler-mardy, visualsfm-mardy, openmvs-mardy, cmvs-mardy, mvs-texturing-mardy, mve, openmvs

Utilisation de VSfM

http://ccwu.me/vsfm/doc.html#usage

1. ajout des images "File->Open Multi Images"
2. feature detection, pairwise image matching "SfM->Pairwise Matching->Compute Missing Match"
3. reconstruction non dense "SfM->Reconstruct Sparse"
4. reconstruction dense "Sfm->Reconstruct Dense".

Utilisation de MVE

https://github.com/simonfuhrmann/mve/wiki/MVE-Users-Guide

1. makescene -i <image-dir> <scene-dir>
2. sfmrecon <scene-dir>
3. dmrecon -s2 <scene-dir>
4. scene2pset -F2 <scene-dir> <scene-dir>/pset-L2.ply
5. fssrecon <scene-dir>/pset-L2.ply <scene-dir>/surface-L2.ply
6. meshclean -t10 <scene-dir>/surface-L2.ply <scene-dir>/surface-L2-clean.ply

Sommaire intermédiaire

Pistes :

- `http://logiciels.ign.fr/?Micmac`
- `http://opensourcephotogrammetry.blogspot.com/`
- `https://www.behance.net/gallery/17168641/3D-Reconstruction-Images-to-3D-model-Free-App`
- `https://en.wikipedia.org/wiki/Comparison_of_photogrammetry_software`
- `reconstructme.net`
- `http://www.meshlab.net/`
- `https://en.wikipedia.org/wiki/CloudCompare`
- `https://en.wikipedia.org/wiki/PCL_(Point_Cloud_Library)`

Des API de développement :

- openMesh `https://www.openmesh.org/`
- CGAL `https://www.cgal.org/`
- Point Cloud Library `http://www.pointclouds.org/`
- OpenCV `https://opencv.org`
- Assimp `http://assimp.org/`
- Open3D `http://www.open3d.org`
- freeGlut `http://freeglut.sourceforge.net/`
- OpenGL Khronos `https://www.khronos.org`

Choix de l'objet :

- taille
- mélange de surfaces planes et gauches
- complexité (trous, anses, variations de surfaces, angles aigus)
- animable /composable
- objet connu : cube, sphère (pour les comparaisons et une évaluation des erreurs commises)

Que vérifier ?

- pas de points extrêmes dans le nuage de points
- triangles, arêtes, points "bien formés"
- triangles, arêtes non recouvrants, non vides (manifold)

Plus dur :

- bruit ?
- inversion intérieur/extérieur ?
- respect des trous et des arêtes ?

Sommaire intermédiaire

Étape 1 : numérisation

Pour la prochaine séance :

1. critiquer les 3 méthodes de numérisation (photogrammétrie, Laser et à temps de vol) : moyens à mettre en place, résolution (et "loi" d'évolution de cette résolution), données d'entrée et de sortie, coûts,
2. choisir un ou plusieurs objets à numériser, qui soient compatibles (en taille et complexité), accessibles (sur plusieurs séances),
3. numériser ces objets
4. faire un état du nombre de faces, de la qualité des surfaces ou des nuages de points obtenus, un histogramme des angles dièdres entre faces, la répartition des triangles "a peu près" équilatéraux et au contraires des triangles trop fins, un "calcul" des courbures (*meshlab*)
5. construire une visualisation simple (freeglut, OpenMesh ou Assimp, caméra+rotations sur 3 axes de l'objet)