Large-scale and high-quality Multi-view stereo

Hoang-Hiep Vu

ÉCOLE DES PONTS PARISTECH - ENSAM CLUNY

ÉCOLE DOCTORALE MSTIC, UNIVERSITÉ PARIS EST

THÈSE

presenté à l'École des Ponts ParisTech, France

pour l'obtention du grade de

Docteur de l'École des Ponts ParisTech

Specialité : Informatique

soutenue par

Hoang Hiep Vu

Stéreo multi-vues à grande échelle et de haute qualité

Directeur de thèse: Renaud Keriven

Co-Directeur : Christian Père

5 Décembre 2011

Jury :

Rapporteurs :	Pr. Edmond Boyer	-	INRIA Grenoble Rhône-Alpes,France
	Dr. Yann Gousseau	-	Télécom ParisTech, France
Examinateurs:	Pr. Jean Ponce	-	École Normale Supérieur de Paris, France
	Pr. Nikos Paragios	-	École Centrale de Paris, France
	Dr. Luc Robert	-	Autodesk, France
Directeur :	Pr. Renaud Keriven	-	Acute3D - École Polytechnique, France
Co-Directeur :	Dr. Christian Père	-	ENSAM Cluny, France

ÉCOLE DES PONTS PARISTECH - ENSAM CLUNY

ÉCOLE DOCTORALE MSTIC, UNIVERSITÉ PARIS EST

PHD THESIS

presented at École des Ponts ParisTech, France

to obtain the title of

Doctor of Science

of École des Ponts ParisTech

Specialty : Computer Science

Defended by

Hoang Hiep Vu

Large-scale and high-quality Multi-view stereo

Thesis Advisor: Renaud Keriven

Co-Advisor: Christian Père

December 5th, 2011

Jury :

Reviewers :	Pr. Edmond Boyer	-	INRIA Grenoble Rhône-Alpes,France
	Dr. Yann Gousseau	-	Télécom ParisTech, France
Examinators:	Pr. Jean Ponce	-	École Normale Supérieur de Paris, France
	Pr. Nikos Paragios	-	École Centrale de Paris, France
	Dr. Luc Robert	-	Autodesk, France
Advisor :	Pr. Renaud Keriven	-	Acute3D - École Polytechnique, France
Co-Advisor :	Dr. Christian Père	-	ENSAM Cluny, France

Remerciements

Cette thèse s'est déroulée de 2008 à 2011 au groupe IMAGINE (CERTIS) à l'Ecole des Ponts ParisTech. Je voudrais tout d'abord exprimer ma gratitude vers mon directeur de thèse Renaud Keriven, pour son encadrement, sa vision scientifique et son encouragement. Sa bonne humeur et optimisme me aussi motive dans cette recherche. Bien qu'il soit occupé avec Acute3D en 2011, il trouve toujours son temps pour m'aider à finir la rédaction.

Cette thèse ne peut finir sans le financement de l'Ecole Nationale Supérieur d'Arts et Métiers de Cluny. Je remercie à Chritian Père pour son accueil chaleureux à l'Abbeye de Cluny en 2009 et 2010. Je remercie Edmond Boyer, Yann Gousseau pour avoir accepté d'être mes rapporteurs et pour leur effort de relire ce manuscrit. Je remercie Jean Ponce, Nikos Paragios et Luc Robert pour participer au jury.

Je remercie également Bernard Vallet pour avoir pris de belles images pour Cluny, ville de Chamonix et Aiguille du Midi. Je suis reconnaissant de Christophe Strecha pour son site d'évaluation de stéréo multi-vues et le jeu des données de Lausanne utilisé dans cette thèse.

Ma recherche s'appuient sur des travaux de Jean-Philippe Pons et Patrick Labatut au laboratoire. Leur intelligence et leur créativité m'impressionnent et me fascinent toujours. Je les remercie pour leur renseignement et ainsi pour la collaboration durant une bonne partie de ma thèse. Je remercie à David Ok pour son aide dans la traduction d'une partie du manuscrit en français. Je voudrais remercier également Hang Si que je ne connais pas encore, pour son programme tetgen.

Je dis mes grands mercis aux tous les membres du laboratoire que partagent des moments inoubliables pendant trois ans. Grâce à eux, IMAGINE m'est toujours un endroit agréable. Je remercie en particulier pour Mme Brigitte Mondou, la secrétaire du laboratoire ainsi pour Mme Sylvie Cach, la secrétaire de l'Ecole Doctorale pour gérer tous les documents administratifs.

Je pense également à mes professeurs et à mes amis vietnamiens, français et d'autres nationalités qui m'ont enseigné et inspiré.

Finalement, je remercie mes parents, ma petite soeur Vân et ma copine Dạ Vi pour leur soutien, confiance et amour.

Titre: Stéreo multi-vues à grande échelle et de haute qualité.

Établissement: Groupe Imagine, laboratoire d'informatique Gaspard-Monge, l'École des Ponts, l'Université de Paris-Est.

Résume:L'acquisition de modèles 3D des scènes réelles trouve son utilité dans de nombreuses applications pratiques, comme l'archivage numérique, les jeux vidéo, l'ingénierie, la publicité. Il existe principalement deux méthodes pour acquérir un modèle 3D: la reconstruction avec un scanner laser (méthode active) et la reconstruction à partir de plusieurs photographies d'une même scène prise dans des points de vues différentes (méthode passive). Si la méthode active permet d'acquérir des modèles avec une grande précision, il est cependant coûteux et difficile à mettre en place pour de grandes scènes extérieures. La méthode passive, ou la stéréo multi-vues est en revanche plus flexible, facile à mettre en oeuvre et surtout moins coûteuse que la méthode active.

Cette thèse s'attaque au problème de la reconstruction de stéréo multi-vues à grande échelle et précise pour les scènes extérieures. Nous améliorons des méthodes précédentes et les assemblons pour créer une chaîne de stéréo multi-vues efficace tirant parti de l'accélération de cartes graphiques. La chaîne produit des maillages de qualité à partir d'images de haute résolution, ce qui permet d'atteindre les meilleurs scores dans de nombreuses évaluations. Aux plus grandes échelles, nous développons d'une part des techniques de type diviser-pour-régner pour reconstruire des morceaux partiaux de la scène. D'autre part, pour combiner ces résultats séparés, nous créons une nouvelle méthode qui fusionne rapidement des centaines de maillages. Nous réussissons à reconstruire de beaux maillages urbains et des monuments historiques précis à partir de grandes collections d'images (environ 1600 images de 5M Pixel).

Mot clés: multi-vues stéréo, reconstruction 3D, reconstruction de surface, méthode variationelle, GPU, fusion des maillages, grande échelle, haute qualité, précision, diviser pour régner, triangulation de Delaunay, tétrahédralisation contrainte de Delaunay, coupe minimale.

Title: Large-scale and high-quality dense multi-view stereo.

Institution: Group Imagine, laboratory of Computer Science Gaspard-Monge, Ecole des Ponts, University of Paris-Est.

Abstract: Acquisition of 3D model of real objects and scenes is indispensable and useful in many practical applications, such as digital archives, game and entertainment industries, engineering, advertisement. There are 2 main methods for 3D acquisition : laser-based reconstruction (active method) and image-based reconstruction from multiple images of the scene in different points of view (passive method). While laser-based reconstruction achieves high accuracy, it is complex, expensive and difficult to set up for large-scale outdoor reconstruction. Image-based, or multi-view stereo methods are more versatile, easier, faster and cheaper. By the time we begin this thesis, most multi-view methods could handle only low resolution images under controlled environment.

This thesis targets multi-view stereo both both in large scale and high accuracy issues. We significantly improve some previous methods and combine them into a remarkably effective multi-view pipeline with GPU acceleration. From high-resolution images, we produce highly complete and accurate meshes that achieve best scores in many international recognized benchmarks. Aiming even larger scale, on one hand, we develop Divide and Conquer approaches in order to reconstruct many small parts of a big scene. On the other hand, to combine separate partial results, we create a new merging method, which can merge automatically and quickly hundreds of meshes. With all these components, we are successful to reconstruct highly accurate water-tight meshes for cities and historical monuments from large collections of high-resolution images (around 1600 images of 5 M Pixel images).

Key words: multi-view stereo, 3D reconstruction, surface reconstruction, variational method, GPU, mesh merging, large scale, high quality, accuracy, divide and conquer, Delaunay triangulation, constrained Delaunay tetrahedralization, graph cuts.

Contents

Chapter 1

Introduction (version française)

Contents

Nous recevons des informations extérieures grâce à nos sens: la vue, l'ouïe, le toucher, l'odorat et le goût. Pour l'être humain, la vue est le sens le plus important qui capte 80 % des informations. Les yeux reçoivent la lumière , et la convertissent en signaux dans les neurones du cerveau. Le cerveau traite les signaux et nous fait percevoir le monde: la luminosité, la couleur de tout ce que nous pouvons voir.

Cette capacité peut se faire avec un seul œil, alors pourquoi en avons-nous deux ? En effet, le système de vision avec 2 yeux fournit la perception de profondeur et l'estimation des distances relatives entre des objets. Avec un seul œil, cette capacité est très limitée (bien qu'un léger déplacement du point de vue, ou un changement de focale puisse aider à avoir une perception de profondeur). Deux yeux donnent des vues un peu différentes, dans lesquelles le cerveau relie des similitudes pour nous donner l'impression de relief. La perception de profondeur à partir de deux vues est appelée la stéréopsie (ou stéréo-vision). Parfois, cette capacité spéciale nous conduit à une estimation inexacte, *e.g.* nous voyons un poisson dans l'eau plus haut que sa position actuelle (à moins que nous soyons aussi dans l'eau), ou nous pouvons regarder un film 3D avec des lunettes particulières à travers lesquelles chaque œil reçoit codes images différentes du film pour reconstruire le relief dans notre cerveau. Les hommes ont inventé de nombreuses machines pour enregistrer ce qu'ils ressentent : les magnétoscopes pour le son, les caméras pour la vue. Les caméras sauvegardent le monde réel en photos de deux dimensions, qui préservent la luminosité et les couleurs originales. Pourtant, ces photos en deux dimensions ne fournissent pas les informations de profondeur de scène prise. Alors, comment 'enregistrer' l'information 3D ?

Les machines les plus sophistiquées et les plus polyvalentes ont été inventées en XX siècle : les ordinateurs. Les ordinateurs sont les machines les plus proches de l'intelligence du cerveau humain. Un domaine dans l'informatique qui étudie

les méthodes pour donner aux ordinateurs la capacité de comprendre la vue, est la vision par ordinateur. La question centrale est : comment les ordinateurs peuvent-ils interpréter et comprendre le monde de manière similaire au cerveau humain ? Au début, les chercheurs ont pensé que ce travail a été assez simple, mais ils se sont vite rendu compte de sa grande complexité.

Malgré de grandes avancées sur cette question grâce à des algorithmes innovants et la puissance croissante des ordinateurs, la vision par ordinateur ne parvient toujours pas à la capacité de la vision humaine. Une question classique et directe dans la vision par ordinateur est : comment faire pour que l'ordinateur puisse percevoir le relief à partir de différents 'yeux' (caméras) ?

La reconstruction de la structure 3D à partir de multiple images dans des points des vues différentes ou la stéréo multi-vues, est plus ou moins équivalente de la stéréopsie de l'humain. Outre son intérêt théorique, la stéréo multi-vues présente son utilité dans de nombreuses applications, telles que:l'archivage numérique, simulations et mesures sur des modèles numériques, jeux vidéo et films, publicité. Depuis plus de 20 ans, nombreuses méthodes ont été proposées sans toutefois répondre entièrement à l'exigence de qualité et de scalabilité récemment posée. Avant le début de cette thèse, la plupart des méthodes de stéréo multi-vues dans la littérature, ne permettait pas de traiter que des images de faible résolution dans un environnement contrôlé en laboratoire. Les autres méthodes ont de résultats dont la précision et la qualité reste encore à désirer. Le but de cette thèse est de produire des maillages 3D de grande échelle et de haute qualité à partir de nombreuses images de haute résolution.

1.1 Conception générale de multi-vues stéréo

La stéréo multi-vues peut se considérer comme le processus inverse de la photographie de scène fixe. Une photographie étant la projection du monde réel en 3D dans le plan d'image de la caméra, la stéréo multi-vues a pour l'objectif de retrouver la géométrie 3D d'une scène à partir d'un ensemble d'images photographiques de cette scène à partir des points de vues différents.

Bien qu'il existe de nombreux modèles de caméra, le modèle standard reste le modèle sténopé. Dans ce modèle, la lumière venant du monde traverse le centre optique de caméra et arrive au capteur d'image directement suivant une ligne droite (Fig. 1.1).

Dans la vision biologique, les humains et les animaux sont capables d'appréhender le relief et la profondeur en relation avec la position et la direction du regard des yeux. De même, la reconstruction 3D à partir de photographies se base sur la connaissance de la position et l'orientation des caméras pour chaque

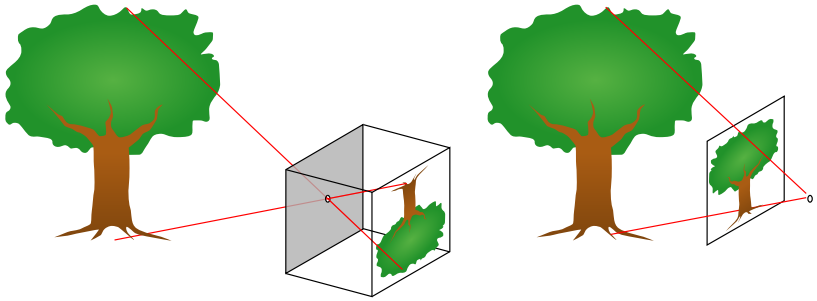


Figure 1.1: Modèle sténopé. Gauche: une configuration de sténopé, l'image est à l'envers, droite: une présentation avec l'image ayant le même sens que l'objet.

photographie. Par conséquent, la stéréo multi-vues nécessite d'abord d'estimer les paramètres de caméras. Une telle estimation s'appelle 'la calibration de caméras', qui dépasse le cadre de cette thèse. Nous invitons le lecteur à se rapporter à l'excellent livre [Hartley and Zisserman, 2004].

Les correspondances des points des différentes images permettent de déduire les positions 3D de l'objet. En effet, si un point 3D A sur la surface de l'objet est projetée à une position a_1 dans le capteur d'un caméra 1 avec le centre d'optique c_1, et une position a_2 dans le capteur d'un caméra 2 avec le centre d'optique c_2 (Fig. 1.2). Alors, connaissant les paramètres des caméras, nous retrouvons les coordonnées du point A en calculant l'intersection des rayons c_1a_1 et c_2a_2. C'est la 'triangulation'.

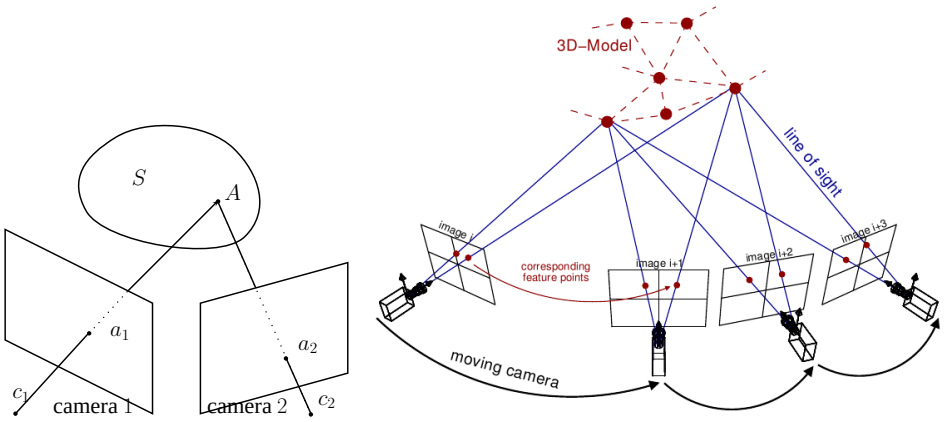


Figure 1.2: Triangulation. Gauche: triangulation à partir de 2 images. Droite: triangulation à partir de multiple images (une image à `http://www.tnt.uni-hannover.de/project/motionestimation/`).

La stereo multi-vues soulève le problème fondamental qui est la mise en correspondance des points dans les images. Les critères de cette correspondance sont très importants dans la stéréo multi-vues et dans d'autres sujets stéréo de vision par or-

dinateur ([Hirschmueller and Scharstein, 2007]). Un critère simple est la différence d'intensité ou de couleur des pixels, mais il est sensible au changement du contraste. D'autres critères ont alors été proposés pour résoudre ce problème, notamment: la corrélation croisée normalisée, ou la description des points clés (des descripteurs comme SIFT [Lowe, 2004], DAISY [Tola et al., 2010]). Cependant, tous ces critères ne reste valable que si la surface conserve la propriété lambertienne: sa lumière est pratiquement invariante au changement de point de vue. Citons quelques exemples de bonnes surfaces lambertiennes: briques, murs, pierres, statues, bâtiments anciens, animaux. Pour les surfaces spéculaires comme le verre, le plastique, l'eau, il reste néanmoins difficile de trouver des correspondances de points de surface basé sur ces critères. C'est pourquoi la plupart des méthodes de stéréo multi-vues supposent que la surface reconstruite est lambertienne.

Figure 1.3: Exemples de surfaces lambertiennes (gauche) and moins lambertiennes (droite). La plupart de méthodes de stéréo multi-vues fonctionnent le mieux pour les surface lambertiennes.

Le problème de stéréo multi-vues le plus simple est l'estimation de la correspondance (ou disparité) à partir de deux ou plusieurs images rectifiées. Nous précisons que la rectification des images consiste à projeter deux (ou plus) images sur un plan commun. Par conséquent, les pixels correspondants des images différentes se trouvent toujours sur la même ligne droite. Depuis l'apparition de l'évaluation quantitative de Middlebury [1] , de nombreuses méthodes d'estimation de la disparité ont été développées et évaluées (voir Fig 1.4 pour un jeu de données). Une étude et analyse en profondeur des différentes méthodes peuvent se trouver dans [Scharstein and Szeliski, 2002].

En raison de sa connexité topologique, un maillage est préféré à une collection

[1] `http://vision.middlebury.edu/stereo/`

Figure 1.4: La disparité des images rectifiées. La troisième image représente la disparité de deuxième image par rapport à la première.

des cartes de disparités. Par ailleurs, les images d'une scène 3D peut être très arbitraires et il est souvent difficile de les rectifier pour calculer leurs cartes de disparité. Plusieurs évaluations, tels que ceux de Middlebury [2] ou de Strecha [3], ont été mis au point afin de comparer des algorithmes différents et d'inciter les chercheurs à créer et à améliorer leurs méthodes.

Figure 1.5: Queques images et la vérité de terrain dans l'évaluation de Middlebury.

1.2 Le sujet de thèse et les contributions

Bien que de nombreuses méthodes de stéréo multi-vues ont été proposées, la plupart restent inapplicables pour reconstruire de larges scènes extérieures en raison du gros volume de données. Les autres n'obtiennent pas des modèles 3D très complets et précis. Le défi de cette thèse est donc de mettre au point et d'améliorer les méthodes de stéréo multi-vues sur les grands jeux de données tout en compte des ressources de calcul limitées. Donc, le sujet de thèse peut s'énoncer de la manière suivante: *On se donne n images d'une scène 3D fixe, prises à partir des caméras calibrées. Calculer un maillage triangulaire de haute qualité de la scène, surtout avec un grand nombre n images d'entrée de haute résolution.*

Après avoir examiné la littérature de stéréo multi-vues dans le chapitre 3, nous écrirons notre chaîne de stéréo multi-vues dans le chapitre 4. Ensuite, nous

[2] `http://vision.middlebury.edu/mview/`
[3] `http://cvlab.epfl.ch/~strecha/multiview/denseMVS.html`

présenterons des algorithmes de type 'Diviser pour Régner' pour adapter cette méthode pour une échelle encore plus grande (chapitre 6, chapitre 7). Pour combiner les résultats partiels, dans le chapitre 5, nous proposerons un algorithme pour fusionner les maillages partielle à un maillage final correct topologiquement et géométriquement.

Plus concrètement, le plan de la thèse est le suivant:

- **Chapitre 3** expose l'état de l'art en stéréo multi-vues.
- **Chapitre 4** présente notre chaîne de stéréo multi-vues qui gère naturellement les scènes de grande échelle et produit des maillages très précis pour des temps de calcul raisonnable. La méthode consiste en 3 étapes principales: la génération de nuage de points quasi-dense, l'extraction d'un maillage qui respecte les contraintes de visibilité, le raffinement variationnel de ce maillage initial pour optimiser la cohérence photométrique. La méthode a été évaluée et nous avons obtenu les meilleurs résultats en terme de la complétude et la précision dans de nombreux jeux de données.
- **Chapitre 5** décrit une nouvelle méthode pour fusionner automatiquement plusieurs maillages séparés à un seul maillage. Basé sur la tétraédrisation contrainte de Delaunay et optimisation de graph-cuts (coupe minimale), la méthode est robuste, efficace et performante pour fusionner de centaines de maillages. Nous allons présenter également une variation qui combine des maillages reconstruits à partir d'une partition d'une boîte englobante, qui sera utilisée dans les chapitres suivants.
- **Chapitre 6** adapte une méthode de reconstruction de surface qui respecte la contrainte de visibilité pour un ensemble très grand de points. En utilisant une représentation à plusieurs niveaux de l'ensemble de point et une approche de Diviser pour Régner, nous réussissons à reconstruire la surface avec la cohérence de visibilité pour les grandes données avec une ressource limitée .
- **Chapter 7** se concentre sur la méthode variationnelle de grande échelle grâce au raffinement de manière séquentielle ou parallèle. Plusieurs expériences de grande échelle seront menées.
- **Chapitre 8** conclut et présente la perspective de cette thèse.
- **Appendice A** décrira les connaissances de base pour cette thèse: la triangulation (constrainte) de Delaunay, qui est utilisé pour la reconstruction de surface et le fusionnement des maillages, coupe minimale, le kd-tree pour la partition de l'espace et des maillages.

- **Appendice B** affichera des résultats supplémentaires de notre méthode dans d'autres jeux de données.

Dans cette thèse, les contributions sont les suivantes:

- Une méthode innovante stéréo multi-vues de haute qualité, capable de reconstruire les modèles 3D de grandes scènes extérieures pour des temps de calcul raisonnable.

- Une nouvelle méthode qui fusionne automatiquement et rapidement un grand nombre des maillages séparés à un maillage correct topologiquement.

- Des algorithmes de type Diviser pour Régner dans la reconstruction de surface et dans le raffinement de maillage de grande taille, pour reconstruire de grands modèles de à la fois complets et précis.

Pendant le projet, nous avons implémenté et optimisé notre méthode en C++, avec la bibliothèque CGAL [4] et OpenGL [5]. Ce projet a permis de reconstruire plusieurs bâtiments et monuments, notamment l'Abbeye de Cluny, pour son 1100 ème anniversaire dans la collaboration avec le projet Gunzo [6].

Cette thèse est financée par l'École des Ponts ParisTech (ENPC) et l'École Nationale Supérieure d'Arts et Métiers de Cluny (ENSAM Cluny).

1.2.1 Liste de publications

Papiers de journal

- H. H. Vu, P. Labatut, R. Keriven and J.-P. Pons. High accuracy and visibility-consistent dense multi-view stereo. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 2011 (accepté, à être apparu).

- F. Lafarge, R. Keriven, M. Brédif, and H. H. Vu. A Hybrid Multi-View Stereo Algorithm for Modeling Urban Scenes. Sousmis in IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI).

Papiers de conférence

- H. H. Vu, R. Keriven, P. Labatut, and J.-P. Pons. Towards high-resolution large-scale multi-view stereo. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2009.

[4] `http://www.cgal.org/`
[5] `http://www.opengl.org/`
[6] `http://cluny-numerique.fr/`

- F. Lafarge, R. Keriven, M. Brédif, and H. H. Vu. Hybrid multi-view Reconstruction by Jump-Diffusion. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2010.

Chapter 2

Introduction

Contents

We receive the external world information through our senses : sense of sight, hearing, touch, smell and taste. For the human being, it is said that the sense of sight is the most significant one which captures 80% of the information. The eyes receive light from the world, and convert it to signals in neurons of the brain. The brain treats the signals and makes us see the world: the brightness, the color of everything that we can see.

This capacity can be done with just one eye, then why do we need 2 eyes? Indeed, the vision system with 2 eyes provides the depth perception and the estimation of the relative distance between objects. With only one eye, this capacity is severely limited (moving the view slightly or adjusting the eye's focal can also help to have a depth perception). Two eyes provide slightly different views from which the brain matches the similarities to give us the impression of depth. This perception of depth from two views is called stereopsis (or stereovision). Sometimes, this capacity leads us to wrong estimation, *e.g.* we see a fish in the water higher than its correct position (unless we are also in the water), or we can watch a 3D movie with special glasses through which each eye receives different images of the movie, to get a depth perception in the brain.

Men have invented many machines to record what they sense: recorder for the sound, cameras for the vision. The cameras save the 'look' of the world into two-dimension photographs that preserve the original brightness and colors. However, those two-dimension photographs do not provide the depth information of the taken scene. Then, how to 'record' 3D information?

The most sophisticated and versatile machines were invented in XX century: computers. Computers are the machines that are nearest to the human brain capacity. A branch of computer science that studies how to make computer understand visual input like images, videos, is computer vision. The main question is how

computers can interpret the world similar to the human brain. In the beginning of computer vision, people thought that this work was easy, but they soon realized how complicated it would be. Despite significant progress due to innovating algorithms and increasing computation power, computer vision still cannot match human vision system for even simple tasks. A fairly straightforward and classic question in computer vision is: how to make computer create 3D information from different 'eyes' (cameras) ?

The reconstruction of 3D structure from multiple images in different views, or multi-view stereo, is more or less the equivalent of human stereo-vision. Beside its theoretical interest, multi-view stereo has many practical applications such as digital archives, simulation and measure over models, game and entertainment industries, tourism, advertisement. For more than 20 years, many methods have been proposed. Nevertheless, most of them only handle low resolution images in controlled laboratory environments. They do not response to the requirement of quality and scalability for large scale. The few others have their results' quality somehow limited. This thesis goal is to solve this problem and to produce high quality large scale 3D models in multi-view from many high-resolution images.

2.1 Overview of Multi-view stereo

Multi-view stereo can be considered as the inverse process of taking photographs of a fixed scene. While a photograph is the map of the 3D scene into a 2D domain through a camera, multi-view stereo has the goal of recovery of this scene from its multiple photographs in different points of view.

While there are many camera models, the standard model is a the pinhole camera (Fig. 2.1). In this model, the light from the world traverse the optical center and meets the image sensor following a straight line.

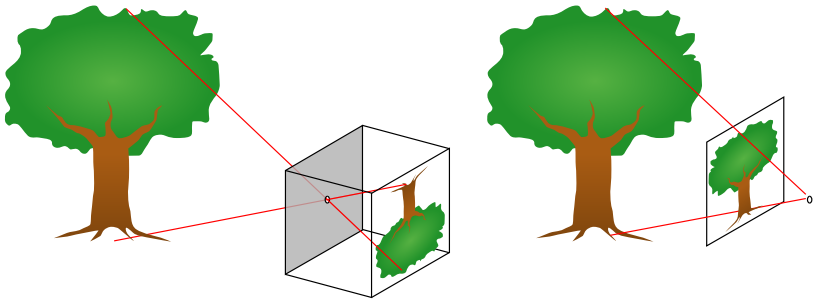


Figure 2.1: Pinhole camera model. Left: a configuration of a pinhole camera, the image is upside down, right: a common presentation with the image has the same orientation as the object.

Men and animals estimate the 3-dimension shapes and depth in relation with the position and the direction of their eyes. Likewise, the 3D reconstruction from photographs (images) relies on the knowledge of position and orientation of the cameras. Therefore, the multi-view stereo usually goes after the estimation of cameras parameters. Such estimation is called 'camera calibration', which is not in the scope of this thesis. We refer interested readers to an excellent textbook [Hartley and Zisserman, 2004].

Matching pixels through multiple images taken from different known views provide 3D positions of the object. Effectively, a 3D point A on the surface of the object is projected to a position a_1 in the captor of a camera 1, and position a_2 in the captor of a camera 2 (Fig. 2.2). Then knowing the camera parameters, such as their optical centers c_1 and c_2, we can compute the position of point A as the intersection of two rays c_1a_1 and c_2a_2. It is called the 'triangulation'.

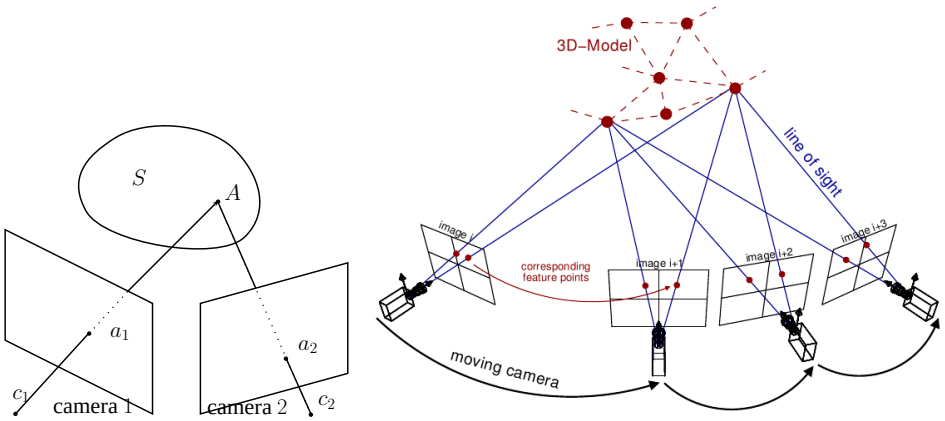


Figure 2.2: Triangulation. Left: triangulation from 2 images. Right: triangulation from multiple images (image from http://www.tnt.uni-hannover.de/project/motionestimation/).

A key component of Multi-view stereo algorithm is the matching the similarity through the images. The criteria for this matching are very important in multi-view stereo and other subjects of computer vision ([Hirschmueller and Scharstein, 2007]). A simple criterion is the difference of intensity or color of pixels, however, it is sensible for light changing. Other popular criteria are then proposed to solve this problem, in particular: normalized cross correlation, or description of key points (descriptors SIFT [Lowe, 2004], DAISY [Tola et al., 2010]). However, those criteria are good only if the scene surface has the Lambertian property: the luminance is almost invariant regardless of the point of view. Good Lambertian surface, for example, are bricks, walls, stones, statues, old buildings, animals. For specular surface like glass, smooth plastic, water, it is difficult to match pixels based on these criteria. That is why most multi-view stereo methods, include our method,

assume that the reconstructed scene is Lambertian.

Figure 2.3: Examples of Lambertian surface (left) and less Lambertian surface (right). Most multi-view stereo methods work best for Lambertian surface.

The simplest form of multi-view stereo is the estimation of correspondence (disparity) from two or more rectified images. We precise that image rectification is the process to project two or more images onto a common image plane. Therefore, the corresponding pixels of different images always lie on the same line. Since the testbed of Middlebury appeared [1], many methods of disparity estimation have been developed and evaluated (see Fig. 2.4 for one data-set). A survey and analysis of different methods can be found in [Scharstein and Szeliski, 2002].

Figure 2.4: Disparity of rectified images. The third image displays the disparity measure of the second image with the first one.

Because of its topological connectivity, a watertight mesh model is preferable to a collection of disparity maps. Moreover, images of a 3D scene may be highly arbitrary so that it is difficult to rectify them to estimate their disparity maps. Several benchmarks (Middlebury [2], Schetra's large scale data-sets [3]) have been set

[1] http://vision.middlebury.edu/stereo/
[2] http://vision.middlebury.edu/mview/
[3] http://cvlab.epfl.ch/∼strecha/multiview/denseMVS.html

up to compare different algorithms and encourage researchers to create and improve their methods.

Figure 2.5: Some images and the ground truths of Middlebury multi-view benchmark.

2.2 Thesis subject and contribution

While a lot of multi-view 3D methods have been created, the nature of outdoor scenes and big quantity of input images has made most of them impossible to apply. Other methods do not provide enough complete and accurate results for large-scale data-sets. The challenge of this thesis is to create, and improve the quality of multi-view stereo methods in order to handle large-scale data-sets within a limited computing resource. Therefore, the subject of the thesis can be stated as: *Given n images of a fix object/scene 3D taken from n calibrated cameras, compute a high-quality 3D triangular mesh of the object/scene, especially with a big number n of images of high resolution.*

After reviewing multi-view stereo literature in chapter 3, we will describe our multi-view stereo pipeline in chapter 4. Next, we will present some Divide and Conquer approaches to adapt this method for even larger scale (chapter 6, chapter 7). In order to combine partial results, chapter 5 will study how to merge partial meshes to a topologically and geometrically correct final mesh.

More concretely, the outline of the thesis is as follows:

- **Chapter 3** will review briefly the state-of-the-art of multi-view stereo.
- **Chapter 4** will present our multi-view stereo method that naturally handles large-scale open scenes, while providing highly accurate reconstructions within a reasonable time. The method consists in 3 main steps: the generation of quasi-dense point cloud with standard passive multi-view stereo techniques, the extraction of a mesh that respects visibility constraints, the variational refinement of this initial mesh to optimize its photo-consistency. The method has been tested and achieved best scores in term of completeness and accuracy in various data-sets.

- **Chapter 5** will describe a new method to merge automatically many separated meshes (in the same coordinate) into a single mesh. Based on constrained Delaunay tetrahedralization and graph cuts optimization, the method is robust, effective and efficient to merge hundreds of input meshes. We will also present its variation to combine meshes built within a partition of a bounding box, which will be useful for next chapters.

- **Chapter 6** will adapt a method of visibility-consistent surface extraction for large input points. Using a multi-level representation of point set and a Divide and Conquer paradigm, we succeed to build the visibility-consistent surface for large point set within limited computing resource.

- **Chapter 7** will focus in large scale photometric variational multi-view stereo (in serial and parallel manners). More large-scale experiments will be conducted.

- **Chapter 8** will conclude and describe the perspective of the thesis.

- **Appendix A** will describe background knowledge for this thesis: Delaunay and constrained Delaunay triangulation which are used for surface reconstruction and merging algorithm, graph cuts, kd-tree for some space partitioning procedures.

- **Appendix B** will display supplement results of our multi-view stereo methods in more outdoor data-sets.

The thesis contribution could be resumed as:

- State-of-the-art reconstruction multi-view stereo method, that is able to reconstruct beautiful 3D model of large outdoor scene in a reasonable time.

- A new automatic and fast merging method to combine many separated meshes into a single, topologically correct mesh.

- A Divide and Conquer paradigm for surface reconstruction, that respects visibility constraint from large point set. A Divide and Conquer variational approach to refine with high-quality large model.

During this project, we have implemented and optimized our methods in C++, with CGAL [4] , OpenGL library [5]. The project successfully reconstructs many urban

[4] `http://www.cgal.org/`
[5] `http://www.opengl.org/`

buildings and monuments, especially the Abbey of Cluny, for its 1100th anniversary celebration and in collaboration with the project Gunzo [6].

This thesis work is funded by École des Ponts ParisTech (ENPC) and École Nationale Supérieure d'Arts et Métiers of Cluny (ENSAM Cluny).

2.2.1 List of publications

Journal papers

- H. H. Vu, P. Labatut, R. Keriven and J.-P. Pons. High accuracy and visibility-consistent dense multi-view stereo. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 2011 (accepted, to be appeared).
- F. Lafarge, R. Keriven, M. Brédif, and H. H. Vu. A Hybrid Multi-View Stereo Algorithm for Modeling Urban Scenes. Submitted in IEEE Transactions on Pattern Analysis and Machine (PAMI) Intelligence.

Conference papers

- H. H. Vu, R. Keriven, P. Labatut, and J.-P. Pons. Towards high-resolution large-scale multi-view stereo. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2009.
- F. Lafarge, R. Keriven, M. Brédif, and H. H. Vu. Hybrid multi-view Reconstruction by Jump-Diffusion. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2010.

[6] `http://cluny-numerique.fr/`

Chapter 3

Review on Multi-view stereo

Contents

Multi-view stereo literature is extremely rich that it is difficult to describe all tendencies. They differ not only in algorithms but also in the conditions of inputs and outputs. Some methods can work best in some kind of images (*e.g.* with silhouette information), but work less on others (*e.g.* uncontrolled outdoor images, specular surface). The outputs are usually a triangular mesh, but also a set of primitives (planes, spheres, etc.) designed for urban building, or a zero level of a function. An excellent reference of the domain is [Seitz et al., 2006] which categorized multi-view methods according to six properties: the scene representation,

photo-consistency measure, visibility model, shape prior, reconstruction algorithm and initialization requirements.

There are other ways to classify multi-view stereo. In chapter 2 of the thesis of Gargallo [Gargallo, 2008], he divided multi-view stereo methods in 3 categories: (i) Bottom-Up: from images, detect 3D points whose projections into the images are sufficiently similar and then make a surface from these points, (ii) Top-Down: instead of searching each 3D points, the whole surface is searched to match images and (iii) Hybrid approach that combines the two previous approaches. The short review of Pons in chapter 8 of his thesis [Pons, 2005] focused on methods that used a 3D volume or surface as initial priori shape. In his term, *space carving framework* consists of all methods that labeled voxels as 'empty' or 'occupied' and the surface is the interface between empty and occupied voxels. The other methods use a 2-dimension surface and refine it to match the images (*deformable model*).

In this chapter, we will explore the framework of photo-consistency and regularization measures used in most methods. Next, we will review the transformation from one type of data to another during the reconstruction of 3D scene from input images. We will examine different multi-view stereo methods for the large scale reconstruction problem. In the end, we take a brief overview of other 3D reconstruction techniques in computer vision besides multi-view stereo.

3.1 Common concepts

Image: An image I of size $w \times h$ is a finite set of values $I(x, y) \in \mathbb{R}^d$ where $x \in \{0, \dots, w-1\}$, $y \in \{0, \dots, h-1\}$, $d = 3$ for color image, $d = 1$ for intensity image. For a non-integer position (u, v) in the range of image, $I(u, v)$ is computed as an interpolation of value of pixels around (u, v). Therefore, I can be considered as a function in a continuous domain $[0, w-1] \times [0, h-1]$.

Camera: A popular model of the camera is the pinhole camera. This model means that the 3D point of the model surface, the camera center, and the corresponding pixel are collinear. Because the image captor is considered plat, this pinhole camera assures that a straight line of the world remains a straight line in the image. Mathematically, a pinhole camera is presented as a matrix 3×4, the projection is a formula of projective geometry. With 2 images I_1, I_2 of pinhole cameras, for a pixel p_1 in I_1, we do not need to search all pixels of I_2 to find its match, but only to of a segment in I_2 that is called 'epipolar line'. More descriptions are found in the book [Hartley and Zisserman, 2004].

Depth map/Image range: Each pixel p of an image, corresponding to a point $P \in \mathbb{R}^3$ of the scene surface, the depth of pixel p is the distance between the camera center C to the perpendicular projection of P in the principal direction line of the

camera. The depth map of an image is the set of depth of all its pixels.

Point cloud/Point set: A set of points in $\mathbb{R}^3$ that is a sample of the object surface. Depending on algorithms to create this point cloud, it may contain many points that are far from the actual surface (called outliers). Each point may associate with images from which it is triangulated.

Patch: A patch is 3D rectangle from a surface. Normally it is a tangent patch of the surface. From a patch, we have a 3D point with its normal vector.

Triangular mesh is a polyhedron whose facets are triangles. Triangular mesh is popularly used in computer graphics and game industry, where render and texture projection is easily computed with commodity hardware like video card. Triangular mesh is an economic representation of the object surface. Its main drawback is the difficulty to handle topological change during surface evolution.

3D Volume is a volume that contains the 3D object. Normally it is a rectangular cuboid, but it could be a fill polyhedron like visual hull, or a convex hull of 3D points.

Voxel: is the smallest entity of a 3D volume. If 3D volume is a rectangular cuboid grid, then voxels are inside small cuboids. A voxel of a Delaunay tetrahedralization is a tetrahedron.

Level set: the object's surface is represented implicitly as the zero level set for a scalar function f in $\mathbb{R}^3$. For implementation, the function is typically defined over a volumetric grid, or over a small band near the surface. The main advantage of level methods is any deformation of the surface can be done by modifying the underlying function. Thus, it also handles naturally topological change. However, its has many shortcomings: large memory consumption for the grid that leads to expensive computation, difficulty in tracking correspondence during the surface evolution.

Silhouette: The projection of the object surface on an image. Such projection shape can be easily estimated by segmentation, especially if the background color is homogeneous and quite different to surface color *e.g.* a white statue in a black background.

Visual hull: If we know the silhouette of the object on an image, then the surface belongs to the back projection of the silhouette to space, which is called a cone of silhouette. When the silhouette is available on many images, the object must be contained in the intersection of their cones of silhouette. We call this intersection volume a 'visual hull'.

3.2 Framework in photo-consistency and regularization measures

Multi-view stereo is based on the similarity and the matching among pixels (or sub-pixels) of different images. The similarity of pixels depends on some photo-consistency of their values. Moreover, we expect the depth map of an image is quite continuous (except in occlusion area), and the object surface is rather smooth. The reconstruction then needs another measure, which is called the regularization.

Formulating in Bayesian approach, given $\mathcal{I}$ the set of images, we need to find the surface $\mathcal{S}$ such that the probability $\mathbb{P}(\mathcal{S}|\mathcal{I})$ is maximized. We have: $\mathbb{P}(\mathcal{S}|\mathcal{I}) = \mathbb{P}(\mathcal{S})\mathbb{P}(\mathcal{I}|\mathcal{S})/\mathbb{P}(\mathcal{I})$. The term $\mathbb{P}(\mathcal{I}|\mathcal{S})$ can be considered as data (photo-consistency) term, and $\mathbb{P}(\mathcal{S})$ as regularization term (the probability of the surface occurs independent to images). While not all methods formulate in a Bayesian approach, most of them use the regularization for superior quality results.

3.2.1 Photo-consistency matching

Photo-consistency is the heart of multi-view stereo. We review photo-consistency measure of pixels in images, without or with 3D information such as voxel, surface, patch. First, we introduce some common matching costs function. Second, we investigate two categories of photo-consistency estimation: *scene space* and *image space*, which was described in [Seitz et al., 2006].

Matching cost Most photo-consistency measures evokes matching cost of pixels between 2 (or more) images. Given two images I, J, we estimate a matching cost between pixel $p = p_0$ in I and $q = q_0$ in J. We consider sets of $n-1$ pixels p_i in I around p, and q_i in J $(1 \leq i \leq n-1)$ around q. Normally, a low matching cost means high similarity or high photo-consistency.

Some basic matching cost based on difference of intensity:

- Squared intensity differences: SD $= (I(p) - J(q))^2$.
- Absolute intensity differences: AD $= |I(p) - J(q)|$.
- Sum of squared differences SSD $= \sum_{i=0}^{n} |I(p_i) - J(q_i)|^2$.
- Sum of absolute differences SAD $= \sum_{i=0}^{n} |I(p_i) - J(q_i)|$.

We consider the mean, variance, covariance of intensity around p and q: $\mu_I(p) = \left(\sum_{i=0}^{n} I(p_i)\right)/n, \mu_J(q) = \left(\sum_{i=0}^{n} J(q_i)\right)/n$
$\nu_I(p) = \left(\sum_{i=0}^{n} I(p_i)^2\right)/n - \mu_I(p)^2, \nu_J(q) = \left(\sum_{i=0}^{n} J(q_i)^2\right)/n - \mu_J(q)^2$.
$\nu_{I,J}(p,q) = \left(\sum_{i=0}^{n} I(p_i)J(q_i)\right)/n - \mu_I(p)\mu_J(q)$.

Another variation is computing these values by a convolution of image window with a Gaussian distribution.

- Normalized cross correlation (NCC): $\nu_{I,J}(p,q)/\sqrt{nu_I(p).nu_J(q)}$. Usually, we use $1 - NCC$ as the matching cost because we wants low value means high similarity. In some articles, this NCC term is also called zero-mean normalized cross correlation (ZNCC).

The NCC is more robust than SD, AD, SSD, SAD because it tolerates a linear change of intensity: if $J(q_i) = aI(p_i) + b$ then we have: $1 - NCC_{I,J}(p,q) = 0$. On the other hand, it is a more costly. There are other matching cost functions, like Mutual Information (MI), histogram descriptors.

We now suppose have n images and a small continuous 3D representation S (it can be a piece of surface, a voxel, a patch). How can we measure the photo-consistency of S in relation with the images ? We will investigate the answer in two categories described in [Seitz et al., 2006].

Scene space photo-consistency The idea is to project S into the images and measure the mutual agreement of its projections. Some methods, especially which compute voxels photo-consistency, measure the homogeneity of projected pixels in the input images. Space carving in [Seitz and Dyer, 1999] used a function as the average of intensity variance of projected pixels. [Hornung and Kobbelt, 2006b] chose a discrete point sample inside voxels to compute the voxel photo-consistency as the sum of normalized color variances per sample.

Another solution is to compute pair-wise photo-consistency. Using pair-wise NCC is common in case S is a patch, or a voxel with a normal vector. [Furukawa and Ponce, 2008] optimized this photo-consistency to find out the best normal vector and center of surface patches. [Sinha et al., 2007] computed this photometric measure with different normal direction to determine the crossing faces of the surface in a volume. In those methods, a discrete set of points, sampled from the patch, is projected to images to compute the photo-consistency. Voxel photo-consistency in [Vogiatzis et al., 2005], [Tran and Davis, 2006], [Starck et al., 2006] also used pair-wise NCC, however, they did not optimized normal vectors. In [Vogiatzis et al., 2005], [Vogiatzis et al., 2007], the photo-consistency of voxels P (or their center point P) is a function of NCC of fixed windows around its projected pixels. This is not quite accurate because two rectangles of two images are unlikely matching. [Faugeras and Keriven, 1998] used an an approximate tangent plane of each point on the surface to compute the matching cost, which would be integrated over the surface.

Image space photo-consistency It is quite similar to scene space photo-consistency of surface (or patch) with many pairs of images. One difference is we choose sample points over image (pixels) and not on the surface. Another difference is the matching cost is integrated over the image domain.

One image is reprojected to another image by the surface S and the matching cost is computed in a single image domain. Comparing the predicted and measured image yields a photo-consistency measure known as *prediction error* in [Szeliski, 1999], [Hernández and Schmitt, 2004]. Deformable surface methods, such as [Pons et al., 2007], [Gargallo et al., 2007], [Delaunoy et al., 2008] also optimized photo-consistency over image domain. In those methods, more weight of photometric attributes to the parts of the scene that are frequently viewed or occupy large image areas.

This reprojection technique is also widely used in plane-sweeping depth maps reconstruction, first introduced in [Collins, 1996]. For a reference camera, a plane is swept in the reference camera frustum, and its offset follows a geometric sequence between the near and far planes of the camera. Other images are projected to the reference image by this plane then the matching cost is estimated for all pixels. Hence, in each pixel, the photo-consistency measures with other images via all the planes are computed. In Winter-take-all strategy, the pixels memorize the highest score and the associated planes that provide the depth map. Plane-sweeping is particularly suitable for GPU acceleration such as [Kim et al., 2007], [Merrell et al., 2007], [Zach et al., 2007], [Frahm et al., 2010].

3.2.2 Regularization

Data-driven term as photo-consistency is not enough to produce a quality 3D model. From a Bayesian view point, the surface model depends not only on its images, but also on what it is expected to be (regular and smooth). Beside this Bayesian analysis, another reason to add some regularization is that the captured images are more or less noisy. The pixel intensity/color is discretized from the light entering the image sensor, which is a source of noise. Moreover, the camera calibration may not be accurate enough so that we can fully trust photometric measure. Photo-consistency alone does not provide enough accurate and complete information, hence, regularization is necessary.

In addition, multi-view methods which rely on non-global optimization, such as deformable model, can fall to a local optimization surface. The regularization helps to reduce the number of possible local optimization traps and makes the refinement process more robust. In the following, we describe common regularization in 2 presentations of an object surface: depth maps and 3D surface.

Depth maps smoothness We expect the depth from neighbor pixels differs slightly, except in occlusion zone. In fact, this expectation is already used implicitly with window-based photometric matching cost. The matching cost of a pixel is computed with the assumption that neighboring pixels share equal or similar depths. The regularization of depth map is mostly presented within 2 approaches: belief propagation and energy optimization.

- Propagation belief: There are some distinctive pixels which we could find correspondences more correctly than others (key points like SIFT, Harris, Corner points, pixels in highly texture zone). The idea of propagation belief is from those matching seeds, we consider match among the neighborhood pixels around them. The matching cost of the neighborhood pixels is then verified. The pixels with high photo-consistency measure are considered as new seeds and the propagation process continues. While this is the main idea of propagation belief, there are many details left for different methods, such as the threshold of matching cost, the order of propagation, the outlier filter. Some multi-view methods used propagation belief: [Strecha et al., 2003], [Lhuillier and Quan, 2005], [Goesele et al., 2007], [Cech and Sara, 2007], [Furukawa and Ponce, 2008].

- Energy optimization: This approach optimizes an image-based smoothness term that seeks to give neighboring pixels the same depth value. This prior fits nicely into a 2D Markov Random Field (MRF) framework. Some methods globally optimized an energy [Kolmogorov and Zabih, 2002], [Campbell et al., 2008]. Some others used Bayesian formulations, that combine photo-consistency, depth smoothness, visibility in their energy, and then optimized within an EM framework [Strecha et al., 2004], [Gargallo and Sturm, 2005], [Tylecek and Sara, 2009].

Surface smoothness The common regularization used in deformable/variational surface is the minimization of surface area or surface curvature. If the energy is set as the integration of a function (typically photo-consistency) over the surface, a regularization term may not be necessary because this energy already favors small surface. Therefore, some methods, using level-set, such as [Faugeras and Keriven, 1998], [Zhao et al., 2001], [Lhuillier and Quan, 2005], [Jin et al., 2005] did not use a regularization term. Some others still added a regularization term in the energy: [Hernández and Schmitt, 2004], [Sinha et al., 2007], [Furukawa and Ponce, 2008] . One benefice of using smoothness operator is to distribute vertices more equally around the mesh and avoid degenerated facets.

Variational surface method with reprojection error optimization does not have

that self-regularizing energy because the energy is integrated over image domain. Thus, regularization is imperative in this case. [Pons et al., 2007], [Gargallo et al., 2007], [Delaunoy et al., 2008] used weighted area functionals integrated over the surface into their formula (typically a surface area term). Another regularization is a thin-plate energy, that penalizes strong blending, and leads to the bi-Laplacian operator [Vu et al., 2011].

3.3 Transformation steps in multi-view stereo methods

We consider a full multi-view stereo method is a process that transform images information (with their camera matrices) into a watertight surface. During this transformation, many intermediate representations and algorithms could be used. We will describe different methods in intermediate steps: 'images → discrete presentation', 'discrete presentation → surface', 'surface → surface', 'volume → surface'. A discrete presentation consists of point set, set of patches, depth maps (or range images). A volume is a volume containing the scene, like a grid box, or a visual hull. Some transformation like 'discrete presentation → surface', or 'surface → surface', could use a volume in their algorithms, but we do not include them in the 'volume → surface' part. A transformation could be used many time, not necessary consecutively, to improve the quality of the results.

3.3.1 From images to a discrete presentation

Computation of disparity and depth maps from pairs of images is a traditional research topic. Lots of methods have been proposed, particularly in two-frame stereo. Doing an extensive survey of two-frame corresponding is out of scope of this thesis, and we refer interested readers to [Scharstein and Szeliski, 2002] for an excellent taxonomy. Popular techniques in binocular stereo, beside Winner-Take-All (WTA) approach, are global optimization (dynamic programming, graph cuts, markov random field, etc) and propagation belief. Most multi-view stereo compute depth maps quite similarly to binocular stereo. Some used directly binocular stereo by considering each pair of image, rectifying them and estimating disparity [Cech and Sara, 2007], [Bradley et al., 2008].

An efficient WTA approach in multi-view stereo is using plane-sweeping to find out for each pixel, the depth associated with the best photo-consistency measure. It has the advantage of using a multi-core GPU to accelerate the algorithm. Lack of regularization mechanism, this approach usually produces noisy depths, which need to take care afterwards, by using depth fusion [Merrell et al., 2007], [Zach et al., 2007], [Tylecek and Sara, 2010], or by visibility filter [Vu et al., 2011].

Depth map estimation could be formulated as a global optimization problem, combining photo-consistency and regularization. It includes discrete label Markov Random Field optimization [Campbell et al., 2008], graph cuts [Kolmogorov and Zabih, 2002]. Heuristic optimization is found in Bayesian approaches [Strecha et al., 2004], [Strecha et al., 2006], [Gargallo and Sturm, 2005], [Tylecek and Sara, 2009]. In propagation belief methods, the initial seeds whose depth is accurately estimated are key points from a SfM estimation [Goesele et al., 2007], or high photo-consistency patches [Furukawa and Ponce, 2008].

3.3.2 From a discrete presentation to a surface

One crucial step is the transformation of a discrete presentation (depth maps, set of points, patches) to a watertight surface. The reconstruction the surface from depth maps or a point set is called surface reconstruction, which is the central problem of laser rang-scanning acquisition. Many surface reconstruction methods from laser rang-scanning acquisition, could be successfully applied in multi-view stereo, providing that the depth maps or point set are dense and clean enough. On the other hand, any surface reconstruction methods originated in multi-view stereo community, also has a possibility to apply in range finding acquisition.

[Curless and Levoy, 1996] used volumetric integration to cumulate range scan data and estimate the signed distance of voxels. The surface is then extracted as the zero-crossing iso-sufrace from the volumetric grid. It is used in a multi-view method [Goesele et al., 2006] to create a watertight surface. [Zach et al., 2007] developed a total variation regularization and used L^1 data term ($TV - L^1$) as a functional to estimate a more robust distance function. Poisson surface reconstruction method [Kazhdan et al., 2006] is applied in many recent multi-view methods [Goesele et al., 2007], [Furukawa and Ponce, 2008], [Tylecek and Sara, 2010].

Some other works build the Delaunay triangulation of point set and extract the surface based on visibility of line-of-sight: graph cut optimization [Labatut et al., 2007], fast heuristic optimization in [Pan et al., 2009], [Lovi, 2010]. The similar work in [Labatut et al., 2009b] for range data is proven to be remarkably robust to outliers. A very recent work [Jancosek and Pajdla, 2011] improved this method to reconstruct textureless surface.

3.3.3 From a surface to a surface

When we obtain a watertight surface of the scene from a discrete representation, this surface might not have decent quality. The insight is to deform this surface to optimize some quality criteria such as photo-consistency, regularization or distance to a point set.

The first surface deformable (variational) method is [Faugeras and Keriven, 1998], which optimized a photo-consistent measure integrated over a level-set surface. [Zhao et al., 2001] introduced a deformable surface framework to minimize the distance towards an input data set (points, curves, surface patches). Further work is developed in [Jin et al., 2005], [Hernández and Schmitt, 2004], [Lhuillier and Quan, 2005], [Tylecek and Sara, 2010]. Other deformable surface methods minimize reprojection error [Pons et al., 2007], [Gargallo et al., 2007], [Delaunoy et al., 2008], [Vu et al., 2011]. The main difference between these two approaches is the scene space photo-consistency and image space photo-consistency, which has been discussed in sub-section 3.2.1.

3.3.4 From a volume to a surface

One important multi-view tendency is using a volumetric grid, computing photo-consistency of voxels and then extract a surface that minimizes an energy.

A heuristic surface extraction method is space-carving [Kutulakos and Seitz, 2000], in which low photo-consistency voxels are consecutively removed from a volume. The surface is then the interface of remaining voxels. This method relied on 'hard-decision' of each voxel. A wrong removal of a voxel could not be undone and it can lead to further wrong decisions. Therefore, the order of traversal is important. Moreover, it used a global threshold to remove voxels, and the choice of this threshold is often problematic. In consequence, the space-carving method is susceptible to noise and outliers, and typically yields very noisy reconstruction. The hard decision is replaced by 'soft decision' in probabilistic space-carving [Broadhurst et al., 2001]. The order of traversal is improved in [Yang et al., 2003], including a smoothness term for voxels. Nevertheless, space-carving methods generally do not yield accurate reconstruction.

More robust and sophisticated methods proposed global optimization of an energy over the voxels. The most popular choice of optimization is graph cuts, which was used in [Vogiatzis et al., 2005], [Furukawa and Ponce, 2006], [Hornung and Kobbelt, 2006b], [Hornung and Kobbelt, 2006a], [Tran and Davis, 2006], [Yu et al., 2006], [Sinha et al., 2007], [Vogiatzis et al., 2007]. Typically, voxels are considered as nodes of the graph. Two nodes are connected if their two correspondent voxels are neighbors. In [Vogiatzis et al., 2005], the edge capacity is the photo-consistency cost of the two voxels or the point in the middle of voxels (Fig. 3.1). The nodes (voxels) in the border of the volume connect with the sink with infinitive cost, which mean they are outside voxels. If inside voxels are available, they will connect the source with infinitive cost. A balloon force is applied to avoid the resulted surface to be an empty surface, by connect all voxels to the source with some costs. The big

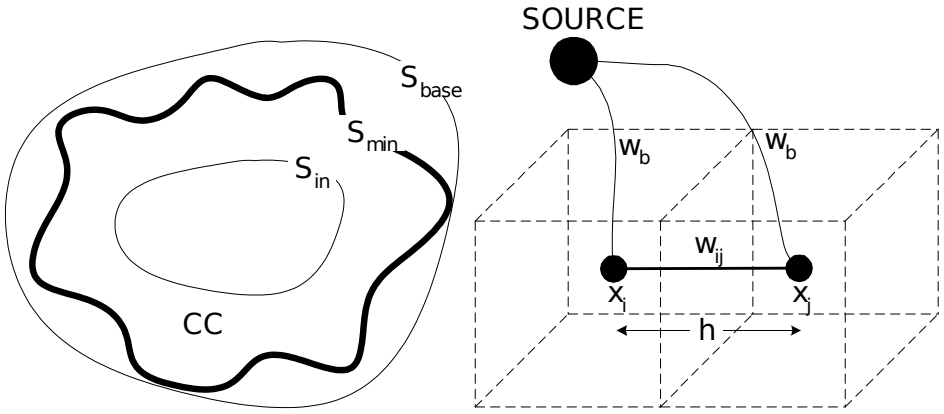


Figure 3.1: Graph cuts in voxels. Figure reproduced from [Vogiatzis et al., 2005].

balloon force creates over-inflated surface and removes thin details. The following work [Vogiatzis et al., 2007] used a better photo-consistency measure. The balloon force can be understood as a shape prior information that could code visibility estimation [Hernández et al., 2007].

[Hornung and Kobbelt, 2006a] replaced volume grid by an octahedral graph structure to reduce memory footprint, which is crucial to process larger data-set. [Tran and Davis, 2006] added surface constraint costs in the graph cuts framework. Different from those work, [Sinha et al., 2007] optimized the cut in an adaptive tetrahedral mesh, that enforced silhouette constraints. [Yu et al., 2006] proposed iterative graph cuts optimization which operated on the surface distance grid, and yielded good results for non-Lambertian objects.

3.4 Large scale multi-view stereo

3.4.1 Multi-view stereo for compact objects

Since the review of [Seitz et al., 2006] and the associated Middlebury evaluation, a lot of research has been focusing on multi-view reconstruction of small objects taken under tightly controlled imaging conditions. This has led to the development of many algorithms whose results are beginning to challenge the precision of laser-based reconstructions. However, as will be explained, most of these algorithms are not directly suited to large-scale outdoor scenes. A number of multi-view stereo algorithms have been proposed that exploit the *visual hull* [Laurentini, 1994].

Many dense multi-view methods rely on this information either as an initial guess for further optimization [Hernández and Schmitt, 2004], [Furukawa and Ponce, 2006], [Hornung and Kobbelt, 2006b], [Hornung and Kobbelt, 2006a], [Starck et al., 2006], [Tran and Davis, 2006], [Vogiatzis et al., 2007], [Yu et al., 2006], as a soft

constraint [Hernández and Schmitt, 2004], [Kolev et al., 2009] or even as a hard constraint [Sinha and Pollefeys, 2005], [Furukawa and Ponce, 2006] to be fulfilled by the reconstructed shape.

While the unavailability of the visual hull discards many of the top-performing multi-view stereo algorithms of the Middlebury challenge [Seitz et al., 2006], the requirement for the ability to handle large scenes discards most of the others, in particular volumetric methods, *i.e.* methods based on a regular decomposition of the domain into elementary cells, typically voxels. Obviously, this approach is mainly suited to compact objects admitting a tight enclosing box, as its computational and memory costs quickly become prohibitive when the size of the domain increases. This includes space carving [Seitz and Dyer, 1999], [Kutulakos and Seitz, 2000], [Broadhurst et al., 2001], [Yang et al., 2003], level sets [Faugeras and Keriven, 1998], [Jin et al., 2005],[Pons et al., 2007], and volumetric graph cuts [Vogiatzis et al., 2005], [Boykov and Lempitsky, 2006], [Hornung and Kobbelt, 2006a], [Lempitsky et al., 2006], [Tran and Davis, 2006] (though [Sinha et al., 2007], [Hernández et al., 2007] propose regular volumetric grid adaptive to photo-consistency measures to push the resolution limit further). Finally, cluttered scenes disqualify variational methods [Faugeras and Keriven, 1998], [Hernández and Schmitt, 2004], [Duan et al., 2004], [Jin et al., 2005], [Lhuillier and Quan, 2005], [Pons et al., 2007], [Delaunoy et al., 2008] that can easily get stuck into local minima, unless a way of estimating a close and reliable initial guess that takes visibility into account is provided.

3.4.2 Multi-view stereo for outdoor scenes

Multi-view stereo methods that have been proven to be more adapted to larger scenes, *e.g.* outdoor architectural scenes, usually initialize the scenes with sparser measurements such as depth maps or point clouds to reconstruct a surface.

The performance of some depth maps based methods [Kolmogorov and Zabih, 2002], [Strecha et al., 2003], [Strecha et al., 2004], [Gargallo and Sturm, 2005], [Strecha et al., 2006], [Goesele et al., 2006], [Goesele et al., 2007] for complete reconstruction however seems to be lower than previously discussed approaches, either as regards accuracy or completeness of the obtained model. This may be due to the merging process and to the difficulty to take visibility into account globally and consistently. While visibility is taken into account to fuse depth maps in [Merrell et al., 2007], the focus on high performance prevents the use of a global optimization. [Zach et al., 2007] proposed a globally optimal variational merging of truncated signed distance maps using a volumetric grid. Another exception could be the work of [Campbell et al., 2008], currently one of the most accurate method according to the Middlebury evaluation, but this method relies on a volumetric

graph cut [Hernández et al., 2007] that cannot handle large-scale scenes.

[Furukawa and Ponce, 2008] proposed a very accurate reconstruction that generates and propagates a semi-dense set of patches. This method has shown impressive results but relies on filtering and expansion heuristics to process a set of oriented patches. The surface reconstruction step that converts the set oriented patches into a mesh, is done by applying the Poisson surface reconstruction [Kazhdan et al., 2006], which requires dense and uniformly sampled point clouds and does not handle visibility issues. Finally, the obtained mesh has to be refined using a mesh evolution. This method has been tested on the data sets provided by Christoph Strecha *et al.* [Strecha et al., 2008], the only available evaluation that allows comparison on large outdoor scenes (to our knowledge). It obtained the best results at the moment of its publication. More recently, [Tylecek and Sara, 2010] used depth map fusion, then refined camera center and mesh refinement, which obtained high accuracy but still lacked the completeness. [Salman and Yvinec, 2009], using our point clouds, achieved nice completeness of the scenes. The method presented in [Jancosek et al., 2009] which aimed to a scalable multi-view stereo, sacrifice the mesh quality and topology for the scalability. This method was designed to have the running time linear of the surface scene.

Our method [Vu et al., 2009], [Vu et al., 2011] will be presented in chapter 4. This method is able to reconstruct accurate large scale open-door scenes from hundred images of 3 MPixel images. However, because it uses a global optimization over a Delaunay triangulation kept in in-core memory, it is not scalable for larger scenes. We will treat this shortcoming in chapter 6. Its variational step, which stores the mesh and images in memory during the photometric optimization, still can handle large mesh in a piecewise manner (a controlled partition process will be described in chapter 7). Different to other large scale multi-view work [Furukawa et al., 2010], [Jancosek et al., 2009], our methods produce a *single* water-tight mesh for connected scenes (not just a union of meshes).

3.4.3 3D reconstruction on Internet scale

Recent progress of Structure from Motion (SfM) and multi-view methods allow researchers handle larger collections of images of a given site available on Internet. The challenge is how to calibrate thousands, even millions of images and how to reconstruct 3D scene from these images within reasonable time. The standard way of calibration is to use bundler adjustment to estimate SfM of all these images [Snavely et al., 2008a], or its skeletal graph to reduce the computing cost [Snavely et al., 2008b]. A preprocessing to remove redundant images and fast matching before calibration could be useful to accelerate the calibration [Frahm et al., 2010]. Implemen-

tations are also taken into account to exploit parallel computing on a cluster [Agarwal et al., 2009] or a single computer with many CPU and GPU cores [Frahm et al., 2010]. For 3D reconstruction, [Goesele et al., 2007] computes depth maps that form a point clouds and water-tight mesh using Poisson surface reconstruction. [Furukawa et al., 2010] partitioned the cameras in view clusters to run a multi-view stereo of choice for each cluster. Taking into account the known vertical of urban scenes, [Frahm et al., 2010] performs GPU-accelerated plane sweeping, and then extracts the polygonal mesh. However, there is no qualitative benchmark on Internet scale in our knowledge and it may be difficult to create one.

Most of those methods are limited to SfM and they only reconstruct point cloud of the scenes. In order to create a watertight mesh, a generic approach is to use a surface reconstruction method. Many surface reconstruction methods are able to handle large point cloud such as streaming Poisson surface [Bolitho et al., 2007]. Nevertheless, such technique requires a relative clean and dense point cloud *e.g.* from PMVS method [Furukawa and Ponce, 2008]. In chapter 6, we will examine more closely surface reconstructions for large, noisy point cloud. The method [Frahm et al., 2010] (among others) exploits the vertical nature of urban scenes to create watertight meshes.

We do not target thousands of photos of tourists on Internet in this thesis. The main reason is the accuracy of calibration for a large collection of images. While our methods are designed for large scale data-sets, we do not always use the full-resolution images because the calibration is not precise enough. While it is possible to apply our method as a component for the reconstruction of huge data-sets, the calibration accuracy is the principal obstacle.

3.5 Some related topics

Multi-view stereo does not stand alone but connects with many other fields in computer vision. It needs the structure from motion methods to calibrate the cameras. It competes with other 3D acquisition method in the range finding registration, shape from X, etc. The 3D reconstruction of dynamical scenes is the generation of multi-view stereos with movement. The comprehension of urban buildings, which requires the detection of geometric primitives and semantics, is a higher level of a brute 3D reconstruction. We will make a small introduction, certainly not complete, of those fields. Therefore, we refer interested readers to many excellent textbooks of computer vision as [Hartley and Zisserman, 2004], [Paragios et al., 2005], [Szeliski, 2010] and relevant papers of each field.

3.5.1 Structure from motion

As we have said in the introduction, multi-view stereo usually requires the accurate knowledge of camera's parameters of the input images. There are two types of parameters: internal (intrinsic) and external (extrinsic) parameters. The internal parameters do not depend on camera's position and are characterized by the focal lengths, principal points in the image, skew value, distortion, etc. The external parameters are the position and orientation of the camera in the world coordinate, which contain the rotation and translation parameters.

If we have little knowledge of camera parameters and the 3D objects, we still can estimate both 3D structure and the camera parameters. This problem is called *structure from motion*. A popular and powerful method to solve SfM is *Bundle adjustment*, which is studied thoroughly in many papers and textbooks such as [Triggs et al., 1999], [Hartley and Zisserman, 2004], [Snavely et al., 2008a]. First, the key points of the input images are extracted then are matched together. Second, from the obtained matching, we estimate initial cameras parameters and triangulate matching pixels to obtain 3D points. These parameters are iteratively adjusted to optimize a sum of the squared distance between matching pixels and computed projection of 3D points. In Bundler software [Snavely et al., 2008a], input cameras are added incrementally during the execution. This incremental approach is quite time consuming and is improved by a direct initialization in [Crandall et al., 2011].

3.5.2 Active range finding

One popular technique to reconstruction 3D object is actively to light the object from one source/sensor and record the (diffuse) reflection by another sensor. It is the mechanism of LIDAR (Light Detection And Ranging) scanners, which are designed for small objects acquisition. The problem of reconstruct objects from range images has been treated in depth from early days of computer vision. Many surface reconstruction methods have been created for this purpose.

In comparison with multi-view stereo acquisition, range finding technique is generally more accurate and does not depend on the object's texture. Nevertheless, the device is expensive, he acquisition process is long and tedious, especially for large objects or exterior scenes. Meanwhile, multi-view stereo, as a passive method, is much cheaper, faster and easy to set up. It is then a more convenient method for a large scale acquisition.

A recent application of the active range finding is in the game industry with the release of Kinect of Microsoft. The depth sensor of the device consists of an infrared laser projector combined with a monochrome CMOS sensor, which captures video data in 3D under ambient light conditions.

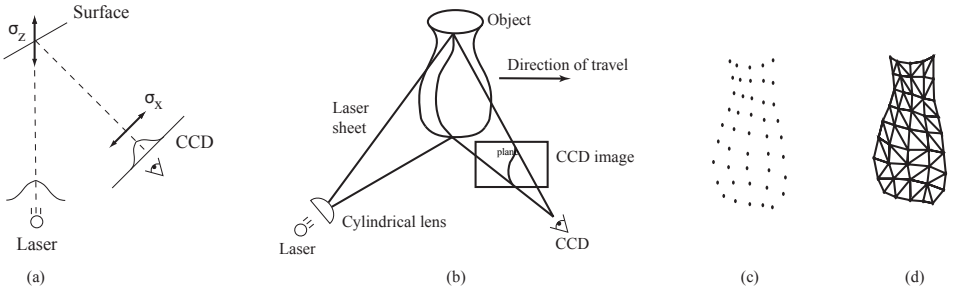


Figure 3.2: Image reproduced from [Curless and Levoy, 1996] (and [Szeliski, 2010]): (a) a laser dot on a surface is imaged by a CCD sensor, (b) a laser stripe is imaged by the sensor, (c) the resulting set of 3D points are turned into (d) a triangulated mesh.

3.5.3 Shape from X

Beside multi-view stereo and active range finding, there are other ways to estimate the shape of objects with information like shading, texture and focus. If we can control and adjust light source, the variation of the object's image provides the object surface orientation such as normal vectors. When the light source is a texture's pattern, the deformation of the pattern wrapped onto the surface also helps to estimate the surface orientation. The camera focus is also a clue of object depth. The amount of blur increases as the object moves away from the camera's focusing distance. By adjusting the camera focus and measuring the blur of images, the depth of the object can be recovered.

3.5.4 Reconstruction of dynamic scenes

In multi-view stereo or structure from motion problem, we suppose the objects are static. This assumption is essential for finding cameras' pose and 3D structure. A slight movement of the object may have a negative impact for the reconstruction. Nevertheless, beside static objects like building, stone, statue, there are lots of objects are in movement like people, cars, animals. The problem is how to recover 3D structure over time with videos or sequence of images over time. This problems takes into account static reconstruction and the movement flows (scene flow in 3D, optical flow in 2D). Lots of methods have been proposed, however, best of our knowledge, the quality of dynamic scenes reconstruction still can not match the static reconstruction. Some methods [Aganj et al., 2009], [Courchay et al., 2009] share some similar ideas with this thesis.

3.5.5 Urban architecture understanding

An architect designs a building not from a polyhedron or a triangular mesh. The form of a typical building is relatively 'simple', and consists almost a set of (vertical

and horizontal) planes or some other primitives like spheres, cones, etc. Different from artistic sculptures, using a dense triangular mesh is unnecessary and heavy for urban buildings. Exploiting the simplicity of urban structures, various methods have been proposed such as the arrangement of planes in [Labatut et al., 2009a], [Furukawa et al., 2009] [Chauve et al., 2010] or combination of geometric primitives in [Lafarge et al., 2010]. The advantage over traditional multi-view stereo is the small storage of result and the better interpretation of the urban scenes.

The higher level of the interpretation of urban buildings is the recognition and semantic information of 3D models. A triangular mesh or a set of primitives of a building does not provide the position of the windows or the walls. Human can recognize the 3D shape of a building and instantly distinguish its components. This capacity can help to design other buildings inspired from existent ones without memorizing their exact shape. This subject is recent in computer vision, with many published works such as [Alegre and Dellaert, 2004], [M uller et al., 2006], [Teboul et al., 2011].

3.6 Conclusion

We have made a short survey in multi-view stereo literature. We try to cover the principal ideas throughout state-of-the-art methods in some key criteria: photo-consistency, regularization, and transformations from one presentation to another. The end of the chapter briefly introduces other fields in computer vision related to multi-views stereo. More discussion about multi-view methods will be presented in the next chapters. The visibility (or occlusion) problem will be discussed further in chapter 6. Some large scale dense multi-views methods will be analyzed in chapter 7.

Chapter 4

Towards large-scale multi-view stereo

Contents

4.1 Introduction

4.1.1 Motivation

The classic problem of scene reconstruction from multiple images finds many practical applications in reverse-engineering, in the game and entertainment industry, and in the digital archives of cultural heritage. However, when high-accuracy reconstructions are required, the reconstruction of outdoor scenes has been traditionally done using range-scanning and a combination of surface reconstruction from point clouds and geometry processing techniques. These methods and the acquisition process are rather complex to set for large-scale outdoor reconstructions, and this

often proves to be time-consuming, expensive and dependent on the scene, particularly when aerial acquisition is required (see for instance the reconstruction of the Bayon temple in Angkor [Banno et al., 2008] which used range finders attached to flying balloons). Providing an image-based reconstruction solution would certainly eliminate most if not all of these drawbacks. This problem has thus always been one of the main goals and an active field of research in computer vision. Recent advances in multi-view stereo methods made this goal closer than ever. In this chapter, the focus is on the dense multi-view stereo problem, *i.e.* , the reconstruction of a surface model from a set of calibrated images where camera calibration is assumed to be accurately known.

4.1.2 Contributions

Our multi-view stereo method consists in a pipeline that naturally handles large-scale open scenes while providing very accurate reconstructions within a very reasonable time. The whole pipeline is designed not to sacrifice accuracy for scalability. Several design choices are made and justified by an analysis of the weak points of other methods. The pipeline contains three main steps:

1. the generation of a quasi-dense point cloud with standard passive multi-view stereo techniques,
2. the extraction of a mesh that respects visibility constraints and is close to the final reconstruction, with a minimum s-t cut-based optimization to fit a surface over the Delaunay triangulation of the points,
3. the variational refinement of this initial mesh to optimize its photo-consistency.

Compared to the preliminary work [Labatut et al., 2007] on robust surface reconstruction from semi-dense point clouds from multi-view stereo matching, the initial point cloud is generated in a denser and more accurate fashion, the surface reconstruction has been adapted to use a more suitable energy similar to [Labatut et al., 2009b]. Finally, the variational refinement uses an energy inspired from work [Pons et al., 2007] but in a lightweight and scalable Lagrangian framework. Our experiments clearly demonstrate its competitiveness on large data sets.

In section 4.2, the different steps of our multi-view stereo reconstruction pipeline are described in details. Implementation aspects are discussed in section 4.3 and section 4.4 presents experiments on a variety of real data sets to demonstrate the potential of our pipeline for reconstructing complex large-scale scenes. Background knowledge such as Delaunay triangulation and Graph cuts are described in Appendix A.

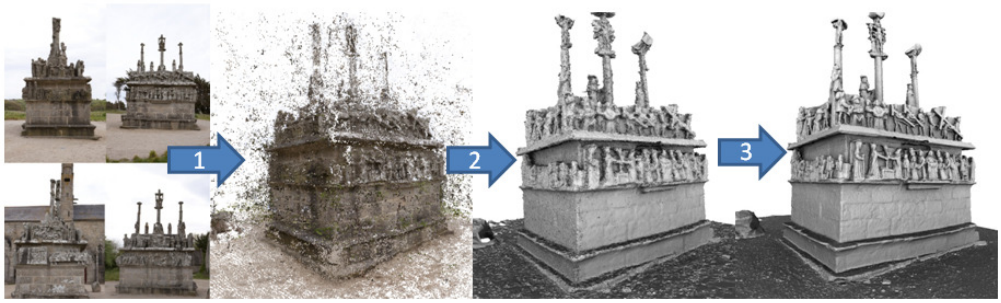


Figure 4.1: Reconstruction pipeline: (1) Generate a points cloud, (2) Extract a visibility-consistency mesh, (3) Refine the mesh with photo-consistency optimization and regularization.

4.2 Multi-view reconstruction pipeline

As shown in Fig. 4.1 and previously announced, our dense multi-view stereo pipeline is composed of three successive stages. Given calibrated cameras associated to the input images, a quasi-dense set of points is first extracted from the images. These points are matched pair-wise between different views: from these matches, a quasi-dense 3D point cloud is generated by reconstructing and optionally merging the triangulated 3D points. This point cloud is then fed to the second stage which builds a Delaunay triangulation from it and then robustly extracts an initial surface from the facets of this triangulation filtering out most of the outliers. Finally, the last step improves the quality of the recovered surface by refining it using a criterion mixing photo-consistency and fairness.

4.2.1 Quasi-dense point cloud

In order to apply the surface fitting of the next step of our reconstruction pipeline, a slightly non-conventional way to generate point clouds from passive stereo is used that favors density over matching robustness. We describe two different but related point cloud generation strategies: one matching interest points in the input images and another using plane sweeping to compute sparse depth maps. We prefer the latter strategy, which is used in all our recent experiments because it generates more points.

Match of interest points First, interest points are located in all the input images. For this purpose, and to capture most of the geometry of the sampled shape, two complementary kinds of interest points are considered: Harris corners which typically lie on "corners" in images and Laplacian-of-Gaussian located at the center of blob-like structures in images. LoG blobs and Harris corners are extracted

at some fixed scale[1] in all the input images. Then, for each potential camera pair (i, j) and for each interest point m_i (of the same type) in the first image I_i of this pair, its best matching point $m_j^\star$ is sought within a small band around the corresponding epipolar line in the other image I_j. The width of this band is fixed and should partially depend on the accuracy of the calibration[2].

The best matching point $m_j^\star$ is the point with the highest matching score against the reference interest point m_i. The neighborhood of a potential match m_j in the image I_j is reprojected in the reference image I_i through a plane parallel to the focal plane of the camera i and passing through the potential reconstructed 3D point (the underlying assumption is that the surface is locally fronto-parallel to the camera i). The matching score can then be estimated in a window around the reference point. Since the choice of an appropriate matching window size is difficult, multi-level matching is used, and the matching criterion is the sum of normalized cross correlations (NCC) for several fixed window sizes[3] (or scale σ) as in [Yang and Pollefeys, 2003].

Furthermore, this best matching interest point $m_j^\star$ is kept only if its matching score is above some threshold and if it is also successfully validated: the original interest point has to be the best matching interest point of its best matching interest point. An initial 3D point can then be reconstructed from the calibration by using standard triangulation optimization [Hartley and Zisserman, 2004].

The final step aggregates the different 3D points. In each image, the 2D Delaunay triangulation of the interest points (of the same type) is computed. This geometric data structure allows to locate efficiently the nearest interest points of a given 2D point. Now, a pair of matched interest points in two different views has given rise to a 3D point by triangulation. By projecting this initial 3D point in the other views, potential other unmatched interest points that are close enough (within a tolerance similar to the half-width of the epipolar band) are located. Closest unmatched interest points are merged with the original pair and a new 3D point (replacing the previous one) is re-estimated from all the interest points. The final result is a set of points each carrying a tuple of views where they were seen. In addition, a confidence value has been assigned to each 3D point, accumulating the photo-consistency scores of all its originating pairs. Obviously, as the whole technique relies on simple greedy or winner-take-all "optimizations", it possibly generates a noisy point cloud with a decent amount of outliers.

[1] in practice, a scale of 2 pixels is used for 6 Mpixel images.
[2] a 3 pixel-wide band is typically chosen for 6 Mpixel images.
[3] 5 levels are used on 6 Mpixel images.

Sparse depth maps While the previous passive stereo approach is general and copes with scenes that have enough texture, it tends to generate lots of outliers and the 3D points are often poorly located. A different passive stereo technique can be devised when strong planar structures are observed as is often the case in architectural scenes.

Initial sparse depth maps are computed between pairs of input images. These depth maps have a downscaled resolution[4] *w.r.t.* the images and are filled using a simple geometric plane sweep with the same thresholded multi-level NCC matching score and winner-takes-all optimization as above. A plane is swept in the reference camera frustum and its offset follows a geometric sequence between the near and far planes of the camera.

These initial depth maps are merged and clusters of points are formed according to their position in the different camera frustums. These clusters are hierarchically split until the bounding boxes of their projections in the images is small enough. A 3D k-D tree [Bentley, 1975] of this clustered initial point set is then build to find efficiently the k nearest neighbors of each point using a large neighborhood[5]. A plane is tentatively fitted to each point's neighborhood with least-squares. Provided the fit is good enough, the point is retained and its position is iteratively refined using the same matching score as above. The final result is the same as what was obtained from interest points: a set of points each carrying a tuple of views where they were seen and an associated confidence. Again, this step still generates a noisy point cloud with a decent amount of outliers but tends to yield better results on architectural scenes (less outliers and noise).

The advantage of the two presented passive stereo techniques lies in the fact that the reprojection and multi-level matching process can leverage the computational resources of common graphics hardware allowing the overall process to be reasonably fast (a few minutes in the data sets of [Strecha et al., 2008] featuring from 8 to 30 images of 6 Mpixel, on an Intel Xeon 3.0GHz CPU with a NVIDIA 260 GTX GPU).

As the reconstruction involves matching points in different images, the corresponding 3D error distribution is complex and cannot be modeled as simply as in the range scanning case. Mismatches are also almost inevitable leading to gross outliers. Depending on the geometry of the cameras and the repetitiveness of texture patterns, these mismatches may even aggregate in structured clusters of outliers producing phantom structures in the point cloud. Another limitation of passive stereo is the highly non-uniform density of samples that depends on the amount of texture on the scene and object. While visibility filtering and expansion techniques combining heuristic-based optimizations have been able to improve the quality of

[4] by a factor 4×4 for 6 Mpixel images.
[5] $k = 25$.

point clouds from stereo as in [Furukawa and Ponce, 2008; Goesele et al., 2007], standard point clouds from multi-view such as the two described acquisition methods have notoriously higher levels of noise and higher ratio of outliers that point clouds acquired with laser range finding.

However, in our case, relying on thresholds and possibly generating numerous outliers is not a serious concern. The only goal of this point cloud from the passive stereo step is to generate enough points so that the following global optimization finds a close enough surface from the tetrahedra facets.

4.2.2 Visibility-based surface reconstruction

The second step of our multi-view pipeline consists in filtering gross outliers from the point cloud and reconstructing an initial surface. These two goals are achieved at once by relying on the Delaunay triangulation described in Appendix A.2 and using a visibility-based formulation to build a surface and discard outliers.

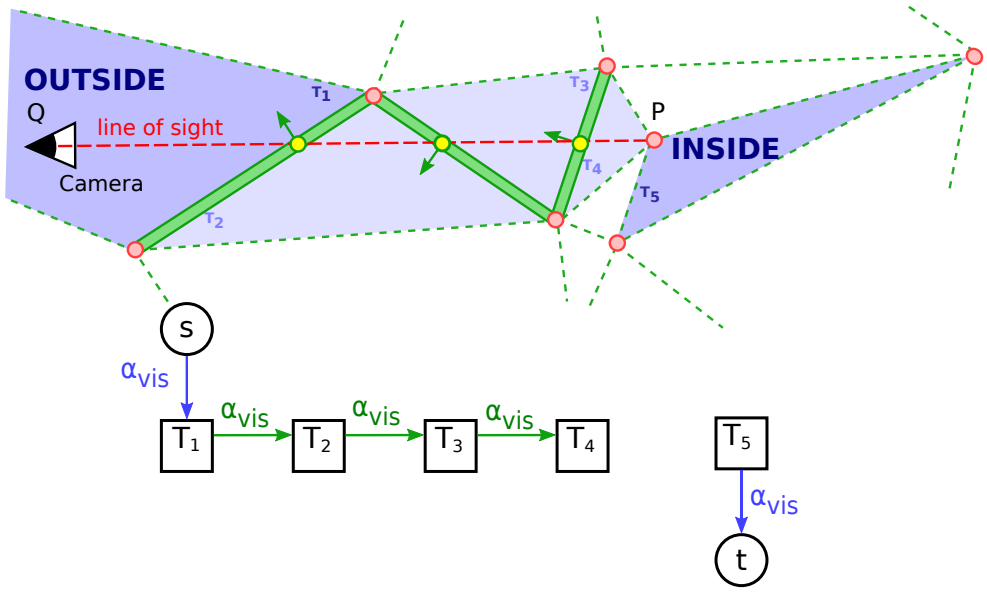


Figure 4.2: ***Visibility and graph construction.*** A line of sight from a reconstructed 3D point traverses a sequence of tetrahedra, the graph construction and the assignment of weights to the tetrahedra and oriented facets.

Optimal tetrahedron binary labeling From the image-based point cloud $\mathcal{P}$ where each point memorizes the two or more images from which it has been triangulated v (as described in the previous section), the 3D Delaunay triangulation of these points is built. Then, the Delaunay tetrahedra are labeled inside or outside the object so that this binary labeling minimizes some energy and finally the surface

is extracted as the set of triangles between inside and outside tetrahedra (called a pseudo-surface in what follows).

Surface visibility A pseudo-surface S^* is sought so as to minimize visibility constraints imposed by the line-of-sight of the acquired points: $S^* = \arg\min_S \mathrm{E}_{\mathrm{vis}}(S, \mathcal{P}, v)$.

A surface should never cross the empty space traversed by the various lines of sight attached to the points. Ideally, one would like to minimize the conflicts of the lines of sight with the surface S induced by the tetrahedron labeling l. This corresponds to the following term:

$$\sum_{P\in\mathcal{P}} \sum_{Q\in v_P} V_{\mathrm{conflict}} \left(l_{T_1^{Q\to P}}, \ldots, l_{T_{N_{[QP]}}^{Q\to P}} \right) ,$$

where $T_1^{Q\to P}, \ldots, T_{N_{[QP]}}^{Q\to P}$ is the ordered sequence of the $N = N_{[QP]}$ tetrahedra crossed from the camera center position Q to the point P (see Fig. 4.2). Since P is a vertex of the Delaunay triangulation, the sequence is terminated before the tetrahedron lying behind P as shown in the upper part of Fig. 4.2. Each oriented facet $F = (T_i^{Q\to P} \cap T_{i+1}^{Q\to P})$ for $i \in [1, N-1]$ is intersected by the line segment $[QP]$. To cast as a minimum $s-t$ cut problem, we penalize the number of misalignments of tetrahedra's label and define V_{conflict} as (we drop the notation $Q \to P$):

$$V_{\mathrm{conflict}}\left(l_{T_1}, \ldots, l_{T_N}\right) = \sum_{i=1}^{N-1} V_{\mathrm{align}}\left(l_{T_i}, l_{T_{i+1}}\right)$$

where V_{align} is a simple pair-wise subterm defined for two adjacent cells of the complex (since in the above equation the cells are crossed in that order, they are adjacent to each other) $V_{\mathrm{align}}\left(l_{T_i}, l_{T_j}\right) = \alpha_{\mathrm{vis}}\, \mathbb{1}\left[\, l_{T_i} = 0 \wedge l_{T_j} = 1 \,\right]$ with α_{vis} is a constant *w.r.t.* the labeling but depends on the considered point or line of sight: it is a confidence measure of the point or line of sight. α_{vis} can be linked to the photo-consistency score of the triangulated 3D point.

Since the trivial labeling $l^0 : t \in \mathcal{T} \to 0$ marking all tetrahedra as outside and to which an empty pseudo-surface corresponds, satisfying these constraints, the facts that the point is assumed to lie near the surface and that the camera centers have to be outside have to be considered. $T_1^{Q\to P}$ is the tetrahedron containing the camera and it should be marked as outside. We denote by $T_{N+1}^{Q\to P}$ the tetrahedron behind the point P in the direction of the line of sight and this tetrahedron should be favored as inside. Therefore we add two more terms: $D_{\mathrm{out}}\left(l_T\right) = \alpha_{\mathrm{vis}}\, \mathbb{1}\left[l_T = 1\right]$ and $D_{\mathrm{in}}\left(l_T\right) = \alpha_{\mathrm{vis}}\, \mathbb{1}\left[l_T = 0\right]$.

To this end, $\mathrm{E}_{\mathrm{vis}}$ is the following expression:

$$\mathrm{E}_{\mathrm{vis}}(S,\mathcal{P},v) = \sum_{P\in\mathcal{P}}\sum_{Q\in v_P} D_{\mathrm{out}}\left(l_{T_1^{Q\to P}}\right) \tag{4.1}$$

$$+ \sum_{i=1}^{N_{[QP]}-1} V_{\mathrm{align}}\left(l_{T_i^{Q\to P}}, l_{T_{i+1}^{Q\to P}}\right) \tag{4.2}$$

$$+ D_{\mathrm{in}}\left(l_{T_{N_{[QP]}+1}^{Q\to P}}\right). \tag{4.3}$$

The corresponding weight construction is shown in Fig. 4.2: the s-link of the vertex representing the tetrahedron T_1 is assigned α_{vis}, the t-link of vertex representing the tetrahedron T_{N+1} ($N = 4$ in Fig. 4.2) behind the point P is assigned α_{vis} and each oriented facet crossed by the line of sight from P to Q is also assigned α_{vis}. These weight assignments are accumulated over all lines of sight, and computing a minimum s-t on this graph yields a globally optimal labeling.

One might wonder if alternatives would not be better suited to this problem, *e.g.* , using the D_{out} subterm for all crossed tetrahedra, that leads to a guided ballooning force [Lempitsky and Boykov, 2007], [Hernández et al., 2007]. Without an appropriate regularization term, that energy tends to minimize the number 'inside' tetrahedra in a light-of-sight no matter whether these tetrahedra are adjacent or not. It might lead to a fragmented surface. On the other hand, our visibility term minimizes the number of time the surface cut the light-of-sights, which favors a more regularized surface.

Surface quality As input images are available, an additional photo-consistency term $\mathrm{E}_{\mathrm{photo}}$ may be used to favor surfaces with best matching re-projections in the different views. This can also be implemented within the minimum s-t cut framework [Labatut et al., 2007]. However the resulting point cloud typically might contain millions of points (see Fig. 4.9), the photo-consistency term is quite expensive. Moreover, the visibility term of our energy is very effective to filter out outliers from stereo point clouds. Since the output surface is only used as an initialization for a variational photometric refinement, the photo-consistency term is advantageously replaced with the simple surface quality term $\mathrm{E}_{\mathrm{qual}}$ of [Labatut et al., 2009b] for surface reconstruction from range scans: $\mathrm{E}_{\mathrm{qual}}(S) = \sum_f w_f \mathbb{1}\left[l_{T_1^f} \neq l_{T_1^f}\right]$. This sum is over every facet f in the triangulation, T_1^f and T_2^f the 2 tetrahedra incident to f, $w_f = 1 - \min\{\cos(\phi), \cos(\psi)\}$, where ϕ, ψ are the angles of the facet f with the circumspheres of T_1^f, T_2^f respectively.(Fig. 4.3).

This term penalizes facets unlikely to appear on a densely sampled surface by using a geometric criterion related to the size of the empty circumspheres of a

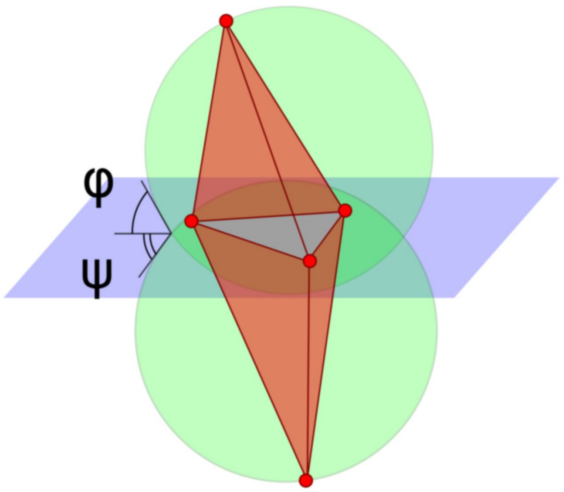


Figure 4.3: ***Surface quality.*** A facet of the triangulation, its two adjacent tetrahedra (red) and their circumspheres (green). Their angles ϕ and ψ with the facet influence the weight this facet will get.

triangle. Support for infinite tetrahedra is also added (tetrahedra with one facet on the convex hull and incident to the infinite vertex). This not only allows the observer to be "inside" the object, but also makes it possible to generate open meshes. This is an important aspect of outdoor scenes.

The energy to label tetrahedra which can be globally minimized with minimum s-t cut, is thus:

$$\mathrm{E}(S) = \mathrm{E}_{\mathrm{vis}}(S, \mathcal{P}, v) + \lambda_{\mathrm{qual}}\, \mathrm{E}_{\mathrm{qual}}(S) \tag{4.4}$$

where $\mathcal{P}$ is the generated point cloud and v the associated visibility sets of the points.

4.2.3 Photometric robust variational refinement

As the initial surface reconstruction method is interpolatory and the point cloud still contains a decent amount of noise, the obtained initial mesh, noted as M^0 is noisy and fails to capture fine details. By using all the image data, this mesh is refined with a variational multi-view stereo-vision approach pioneered by [Faugeras and Keriven, 1998]: M^0 is used as the initial condition of a gradient descent of an adequate energy function. As the mesh M^0 is already close to the desired solution, this local optimization is very unlikely to get trapped in an irrelevant local minimum. The details of the energy function and the optimization procedure are now presented and the improvements over the initial method are justified. This collection of improvements should not be considered as mere implementation details

and all have a strong impact on the accuracy of the final reconstruction.

The initial mesh M^0, as the surface between interior and exterior tetrahedra, may still contain isolated triangles respecting visibility constraint (for example, from false points in the sky, back ground of the scene), or big-size triangles (due to lack of density of points or lack of images in some area). Moreover, it might capture the landscape far from our scene, that we do not need to reconstruct in detail (plus it is impossible to refine this part accurately because of inexact calibration for scenes very far from cameras). For these reasons, we remove these triangles by some threshold of triangle size or number of triangles in an isolated piece, and manually cut unnecessary, far landscape background.

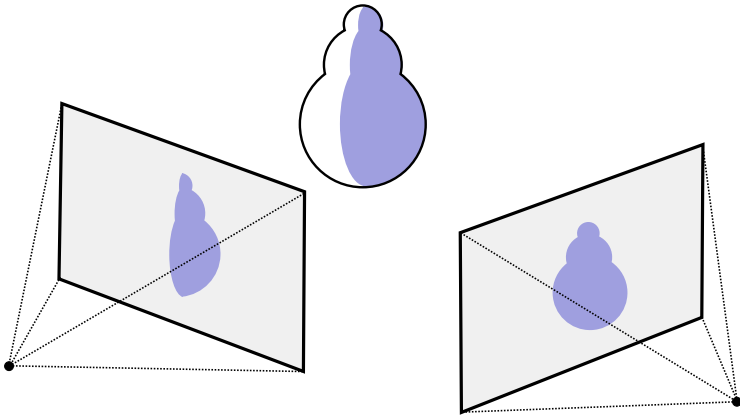


Figure 4.4: ***Reprojection induced by the surface.***

Photo-consistency refinement Let S be the object surface, x a point on S, $\vec{n}$ the normal to S at point x, $g_{ij}(I_i, I_j)(x, \vec{n})$ a positive decreasing function of a photo-consistency measure of the patch $P = (x, \vec{n})$ according to images I_i and I_j, and $v_{ij}^S(x) \in \{0, 1\}$ the visibility of x in these images according to S. The original energy in [Faugeras and Keriven, 1998] is:

$$\mathrm{E}_{\mathrm{photo}}(S) = \sum_{i,j} \int_S v_{ij}^S(x)\, g_{ij}(x, \vec{n})\, \mathrm{d}S \tag{4.5}$$

Instead of this energy, the re-projection error introduced by [Pons et al., 2007] is preferred, namely:

$$\mathrm{E}_{\mathrm{error}}(S) = \sum_{i,j} \int_{\Omega_{ij}^S} h(I_i, I_{ij}^S)(x_i)\, \mathrm{d}x_i \tag{4.6}$$

where $h(I, J)(x)$ is a decreasing function of a photo-consistency measure between images I and J at pixel x (typically the opposite of normalized cross correlation) , $I_{ij}^S = I_j \circ \Pi_j \circ \Pi_i^{-1}$ is the re-projection of image I_j into image I_i induced by S

and Ω_{ij}^S is the domain of definition of this re-projection (see Fig. 4.4), Π_i and Π_i^{-1} are the projection and back projection from an image i to the surface. This energy measures, for each considered camera pair, the dissimilarity between the portion of a reference image corresponding to the projected surface and a portion of another image re-projected via the surface into the reference image.

This summation has several major advantages over the original one:

1. re-projecting I_j into I_i according to S uses the exact geometry of S and does not rely on any approximation of the tangent patch $(x, \vec{n})$,
2. the less a surface element is viewed in a given image, the less it contributes to the energy, and
3. this re-projection can easily and efficiently be computed on graphics hardware with projective texture mapping.

The first point is essential to get an accurate reconstruction: in methods approximating the surface by planar patches, the choice of patch size is a difficult trade-off between robust and accurate photo-consistency. In practice, we set the photo-consistency measure as the opposite of normalized cross correlation (NCC). This measure has the advantage of robustness to noise and light change, that occurs frequently for outdoor images, due to real lighting change and internal image processing inside cameras.

Regularization The original intrinsic energy $\mathrm{E}_{\mathrm{photo}}$ of (4.5) is self-regularizing due to the area-weighted integration over the surface. This is however not the case of (4.6). The energy function $\mathrm{E}_{\mathrm{error}}$ is thus complemented with a surface fairing term $\mathrm{E}_{\mathrm{fair}}$, thin-plate energy that measures the total curvature of the surface. This term penalizes strong bending, not large surface area:

$$\mathrm{E}_{\mathrm{fair}}(S) = \int_S (\kappa_1^2 + \kappa_2^2)\,\mathrm{d}S \tag{4.7}$$

where κ_1 and κ_2 are the principal curvatures of the surface at the considered point. Consequently, the associated gradient flow is exempt from the classical shrinking bias.

4.2.4 Discretization

Many methods in variational multi-view stereovision [Faugeras and Keriven, 1998; Duan et al., 2004; Jin et al., 2005; Lhuillier and Quan, 2005; Pons et al., 2007], and more, generally in computer vision, rely on an *optimize then discretize* approach: an energy functional depending on a continuous infinite-dimensional representation is considered, the gradient of this energy functional is computed analytically, then the obtained minimization flow is discretized.

In contrast, a *discretize then optimize* approach is adopted: an energy function that depends on a discrete finite-dimensional surface representation, here a triangle mesh is considered, and standard non-convex optimization tools are used. The benefits of this approach have long been recognized in mesh processing, but have seldom been demonstrated in computer vision [Hernández and Schmitt, 2004; Slabaugh and Unal, 2005; Delaunoy et al., 2008].

We describe how to compute a discrete gradient flow from a continuous one. The variations of an energy E attached to a surface S can be analyzed with a functional gradient defined as the vector field $\nabla \mathrm{E}$ such that for all vector field v on S, we have:

$$D\mathrm{E}(S)[v] = \left.\frac{\partial \mathrm{E}(S+\varepsilon\, v)}{\partial \varepsilon}\right|_{\varepsilon=0} = \int_S \nabla \mathrm{E}(x) v(x) dx \tag{4.8}$$

If the S is a the triangulated mesh, consisting of n vertices $X_i \in \mathbb{R}^3, i \in [1,n]$, a discrete vector field is defined at the vertices of this mesh by a sequence of vectors $v_i \in \mathbb{R}^3, i \in [1,n]$. Such vector field is interpolated between the vertices over the whole mesh: $v(x) = \sum_i v_i \phi_i$ with $\sum_i \phi_i(x) = 1$ for all $x \in S$ (in the case of triangular facet, $\phi_i(x)$ is the barycentric coordinate corresponding to vertex i if i is one of vertices of a triangle containing x and 0 otherwise).Equation (4.8) becomes:

$$D\mathrm{E}(S)[v] = \sum_i v_i \int_S \phi_i(x) \nabla \mathrm{E}(x) \mathrm{d}x \tag{4.9}$$

This equation naturally shows how to formulate a discrete gradient from a continuous one:

$$\frac{\mathrm{dE}(S)}{\mathrm{d}X_i} = \int_S \phi_i(x) \nabla \mathrm{E}(x) \mathrm{d}x \qquad i \in [1,n] \tag{4.10}$$

As (4.10) shows, the obtained gradient vector at a vertex involves integrals over the whole ring of triangular facets around it (see also [Eckstein et al., 2007], 2.2). This is in strong contrast with a point-wise, and thereby noise-sensitive, dependency on the input data that a late discretization typically causes. A crucial point has to be noted here: this discrete gradient flow may include a significant tangential component driving the vertices at the right places minimizing the energy. For instance, vertices naturally migrate to the object edges if any. This is illustrated by the crisp reconstruction of stair treads in Fig. 4.7.

In what follows, we recall some definitions and results of [Pons et al., 2007] that are the base of our discretization:

Given 2 images: $I, J : \Omega \to \mathbb{R}^d$, let us consider $M(I,J) = \int_\Omega h(I,J)(x)\mathrm{d}x$ as a function of similarity of 2 images. $\partial_2 M(I,J)$ is defined as the derivative of $M(I,J)$

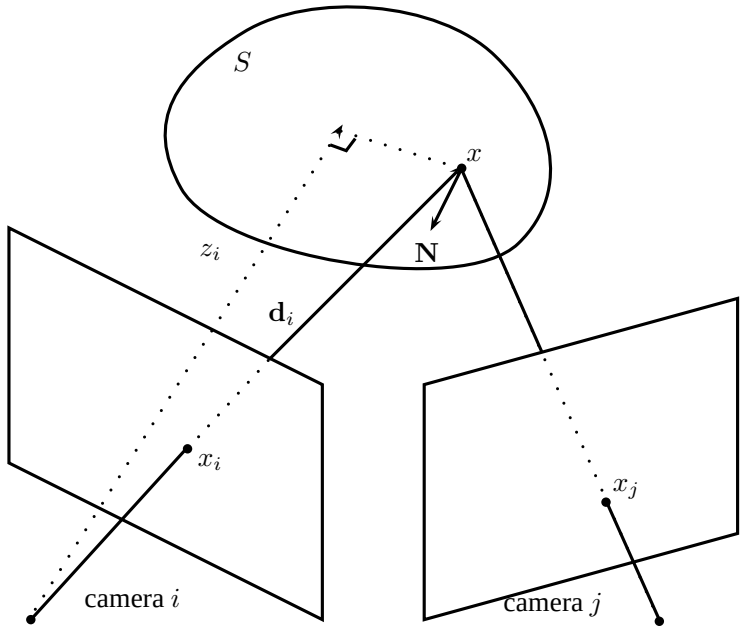


Figure 4.5: ***Some notations in reprojection.***

with respect to the second image, in the sense that for any image variation δJ:

$$\lim_{\varepsilon \to 0} \frac{M(I, J + \varepsilon \delta J)}{\varepsilon} = \int_{\Omega} \partial_2 M(I, J)(x) \delta J(x) dx. \tag{4.11}$$

We note $\mathcal{M}_{ij}(S) = M(I_i, I_{ij}^S)$, thus: $\mathrm{E}_{\text{error}}(S) = \sum_{i,j} \mathcal{M}_{ij}(S)$ and $\nabla \mathrm{E}_{\text{error}}(S) = \sum_{i,j} \nabla \mathcal{M}_{ij}(S)$.

With a point $x \in S$ visible for cameras i and j, we note: $x_i = \Pi_i(x)$, $x_j = \Pi_j(x)$ the projection in image i, j, $\mathbf{d}_i$ the vector joining the center of camera i and x, z_i the depth of x in camera i, $\mathbf{N}$ the outward surface normal at x (see Fig. 4.5). From [Pons et al., 2007] (page 10), we have:

$$\mathrm{d}x_i = -\mathbf{N}^T \mathbf{d}_i \mathrm{d}x / z_i^3. \tag{4.12}$$

$$\nabla \mathcal{M}_{ij}(x) = -\left[\partial_2 M(x_i) D I_j(x_j) D \Pi_j(x) \tfrac{\mathbf{d}_i}{z_i^3}\right] \mathbf{N} \tag{4.13}$$

with M is the abbreviation for $M(I_i, I_{ij}^S)$, $D.$ denotes the Jacobian matrix of a function. The term between square brackets line is a scalar quantity. We note $f_{ij}(x_i) = \partial_2 M(x_i) D I_j(x_j) D \Pi_j(x) \mathbf{d}_i$, then $\nabla \mathcal{M}_{ij}(x) = -f_{ij}(x_i) \mathbf{N} / z_i^3$.

We rewrite (4.10), in dropping the index i of X_i and ϕ_i:

$$\frac{\mathrm{dE}_{\text{error}}(S)}{\mathrm{d}X} = \int_S \phi(x) \sum_{i,j} \nabla \mathcal{M}_{ij}(x) \mathrm{d}x \tag{4.14}$$

$$= -\int_S \phi(x) \sum_{i,j} f_{ij}(x_i) \mathbf{N}/z_i^3 \mathrm{d}x \tag{4.15}$$

$$= -\sum_{i,j} \int_S \phi(x) f_{ij}(x_i) \mathbf{N}/z_i^3 \mathrm{d}x \tag{4.16}$$

$$= \sum_{i,j} \int_{\Omega_{ij}} \phi(x) f_{ij}(x_i) \mathbf{N}/z_i^3 \frac{z_i^3}{\mathbf{N}^T \mathbf{d}_i} \mathbf{N} \mathrm{d}x_i \tag{4.17}$$

$$= \sum_{i,j} \int_{\Omega_{ij}} \phi(x) f_{ij}(x_i) / (\mathbf{N}^T \mathbf{d}_i) \mathbf{N} \mathrm{d}x_i \tag{4.18}$$

where Ω_{ij} is the map of the reprojection from image j to image i via the surface. Therefore, the gradient of each vertex equals the summation weighted (with barycentric co-ordinate) of contribution of all pixels lying in the projection of all the triangles containing this vertex for all pairs of images (i, j).

When the mesh parametrization is close to isometric, the gradient from the complementary thin-plate energy reduces to a simple bi-Laplacian Δ^2. A discrete analog of such simplified thin-plate energy and associated flow, described in [Kobbelt et al., 1998] is used by applying the umbrella operator of [Taubin, 1995] to approximate the Laplace-Beltrami operator. This particular choice has a convenient property of redistributing vertices along the surface, and in particular discourages degenerate triangles.

Balance between photo-consistency and regularization A long-standing issue in variational methods is the proper and automatic balancing between data attachment and smoothing terms. Designing a general solution to this problem is clearly beyond the scope of this paper. A specific strategy is instead proposed that allows to conduct all the following experiments *without adjusting parameters to each data set*. The solution is twofold.

First, the fact that regularization has to be more important where photo-consistency is less reliable is observed, in particular in textureless or low-textured image regions. Consequently, the contribution of camera pair (i, j) at pixel $\boldsymbol{x}_i$ in (4.18) is weighted by a reliability factor $r(x_i) = \min(\sigma_i^2, \sigma_j^2)/(\min(\sigma_i^2, \sigma_j^2) + \varepsilon^2)$, where σ_i^2 and σ_j^2 denote the local variance at $\boldsymbol{x}_i$ in images I_i and I_{ij}^S, respectively, and ε is a constant.

Second, the two terms of the energy function are homogenized: while the data attachment term of (4.6) is homogeneous in an area in pixels, the discrete thin-plate

term is homogeneous in squared world units. After weighting the contribution of each image in (4.6) by the square of the ratio between the average depth of the scene and the focal length in pixels, a scalar regularity weight can be defined whose optimal value is stable across very different datasets. As we previously mentioned, this thin-plate term does not only plays an *a priori* knowledge of the model (Bayesian arguments), but stabilizes the mesh during the refinement by redistributing vertices along the surface.

Mesh resolution The resolution of the mesh is automatically and adaptively adjusted to image resolution: a triangular facet is subdivided if there is one camera pair such that the visible facet projection exceeds a user-defined number of pixels in both images. This threshold is set to 16 pixels in the experiments. A classical one-to-four triangle subdivision scheme is used, which has the advantage of preserving sharp edges. Nevertheless, we believe that other subdivision methods can be used.

4.3 Implementation aspects

Parts of our reconstruction pipeline take advantage of the cheap parallel processing resources available in many consumer-grade graphics card: namely, the computation of the initial quasi-dense point cloud, the computation of the mesh velocity field (the normalized cross-correlation and the image reprojections) also its evolution which are mostly done with a custom combination of vertex, geometry and fragment shaders. The independence of pixels of images in our computation helps our pipeline very adapted to graphics card. Our approach heavily relies on geometric data structures and queries: from the 2D and 3D Delaunay triangulations and its corresponding queries to dynamic meshes. Fortunately the Computational Geometry Algorithms Library (CGAL)[6] [Boissonnat et al., 2000] defines robust and efficient implementations of all the geometric data structures, primitives, queries and traversals needed for our different algorithms. Finally, the max-flow algorithm described in [Boykov and Kolmogorov, 2004] [7] is used to compute a minimum s-t-cut of our specifically designed network graphs.

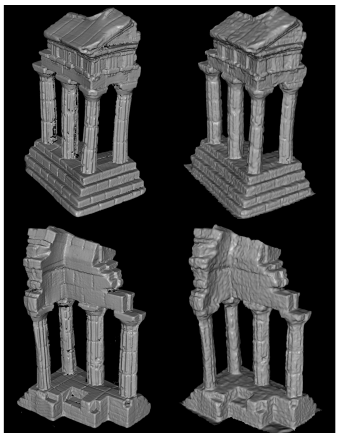
With these implementation advantages, the overall running time is quite reasonable, for example, it takes 45 minutes for the whole pipeline in the dataset Herz-Jesu-P25 provided by Strecha *et al.* [Strecha et al., 2008], consisting of 25 images of resolution 3072×2048. Most of the time being spent either in computing and selecting points when generating the initial point cloud or in the final photometric refinement.

[6] `http://www.cgal.org/`
[7] and implemented in `http://www.adastral.ucl.ac.uk/~vladkolm/software.html`

4.4 Experimental results

4.4.1 Compact objects

As mentioned in the introduction, our reconstruction pipeline does not target small-scale data sets for which the acquisition conditions can typically be easily modified to allow a foreground / background segmentation. Nevertheless, Fig. 4.6 shows the results of our final variational refinement step (from a mesh approximating the visual hull) and evaluation on the Middlebury dense multi-view stereo benchmark of [Seitz et al., 2006]. For the sake of comparison, we have included the results of other methods including the results of our previous level set based method [Pons et al., 2007] and another mesh-based variational approach based on the same energy function as our previous work [Zaharescu et al., 2007]. Our results on the *templeRing* are currently the best both in completeness and accuracy. However, on the *dinoRing*, while a highly complete reconstruction is indeed achieved, our results are less competitive in term of accuracy. This may be explained in the strong lack of texture on this particular data set that makes our photo-consistency measurement less peaked near the ground-truth surface.



	accuracy (at 90%)	completeness (at 1.25mm)
Ours	0.45mm	99.8%
[Campbell et al., 2008]	0.48mm	99.4%
[Furukawa and Ponce, 2008]	0.47mm	99.6%
[Hernández and Schmitt, 2004]	0.52mm	99.5%
[Pons et al., 2007]	0.60mm	99.5%
[Zaharescu et al., 2007]	0.55mm	99.2%

	accuracy (at 90%)	completeness (at 1.25mm)
Ours	0.53mm	99.7%
[Bradley et al., 2008]	0.39mm	97.6%
[Furukawa and Ponce, 2008]	0.28mm	99.8%
[Kolev et al., 2009]	0.43mm	99.4%
[Hernández and Schmitt, 2004]	0.45mm	97.9%
[Pons et al., 2007]	0.55mm	99.0%
[Zaharescu et al., 2007]	0.42mm	98.6%

Figure 4.6: Comparison to ground truth (top images: left column is ground truth, right column is our result) and evaluation results (bottom tables) on the *dinoRing* and *templeRing* data sets of [Seitz et al., 2006].

4.4.2 Outdoor architectural scenes

Provided by Strecha *et al.* [Strecha et al., 2008], the already mentioned data sets consist of outdoor scenes acquired with 8 to 30 calibrated 6 Mpixel images. Ground truth has been acquired with a LIDAR system. The evaluation of the multi-view stereo reconstructions is quantified through relative error histograms counting the percentage of the scene recovered within a range of 1 to 10 times the estimated LIDAR depth standard deviation σ. Dedicated to large-scale objects and fitting perfectly our objective, these sets are particularly challenging, especially the *castle-P19*, a complete courtyard acquired from the inside and where a tractor is placed in the middle, disturbing reconstruction. So far, [Furukawa and Ponce, 2008; Tylecek and Sara, 2009, 2010; Jancosek et al., 2009; Salman and Yvinec, 2009] submitted for all these particular data sets. Some of them appeared after our conference version of this paper [Vu et al., 2009], yet our results still achieve the best at accuracy and completeness in most datasets of this benchmark. Comparisons with the other methods are given in Fig. 4.8, where cumulated histograms clearly show that the proposed pipeline is both more accurate (thanks to the final variational refinement) and complete (thanks to the initial visibility-consistent mesh).

More detailed views of our reconstruction of the *Herz-Jesu-P25* data set are shown in Fig. 4.7. Note how details, topology (*e.g.* columns) and edges (*e.g.* stairs) are precisely recovered while regularization still handles as correctly as possible blurred or untextured parts. Further results are available on the challenge website[8].

4.4.3 Landscape and cultural heritage scenes

The method was tested on an aerial acquisition of the *Aiguille du Midi* summit (data and calibration courtesy Bernard Vallet and Marc Pierrot-Deseilligny respectively). The data set consists of 53 images of 5 Mpixel. Fig. 4.9 shows two of the images, the generated point cloud, the initial mesh M^0 and the final reconstruction. This experiment validates the whole pipeline and the ability to cope with uncontrolled imaging conditions (snow, sun, moving people from one image to another) and a mix of complex and smooth geometries. The variational process is able to recover the top antenna although it is only partially present in M^0. Fig. 4.1 shows results on a data set of 27 images of 10 Mpixel of a sculpted calvary taken from the ground. The cloud has $802K$ points, with many outliers, mainly sky points obtained by matching clouds that have moved between shots. $539K$ of these points are selected for the initial mesh. This mesh is noisy, due to the process of matching interest points that are just approximately view-point invariant. The closer views in Fig. 4.10 show, the final reconstruction ($2,331K$ triangles) is very sharp to capture meaningful details.

[8] http://cvlab.epfl.ch/~strecha/multiview/denseMVS.html

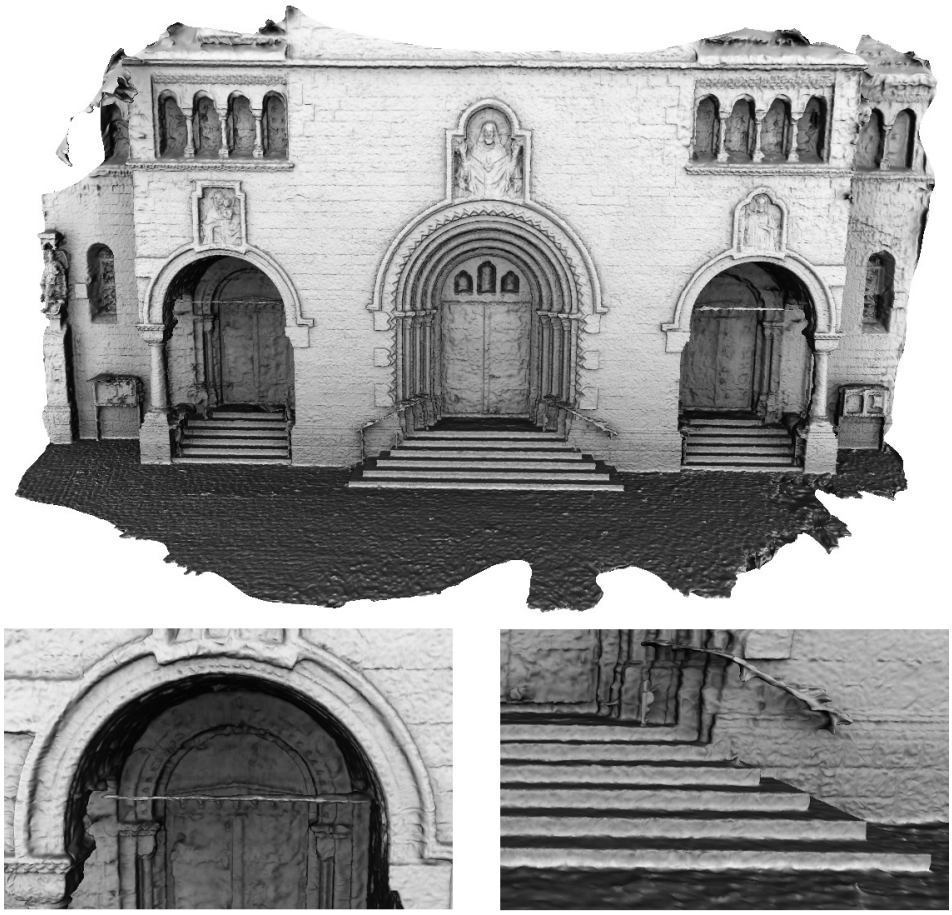


Figure 4.7: Top: overview of our reconstruction of *Herz-Jesu-P25*. Bottom: close-ups on reconstruction details such as the thin metal bars, the facade relief or the staircases.

Fig. 4.11 shows results on a data set of 30 images of 14 Mpixel of Cluny Abbey in France, taken from a balloon in front. Lacking different views, the total scene is not complete, but the final reconstruction proves its great details from the direction of input images. We also tested on an aerial acquisition of *Entrevaux* (Fig. 4.12), consisting of 109 images of 3.1 Mpixel. The final mesh is very complete, captures small details of buildings and cliffs with trees. Note that trees are not suitable for multi-view or mesh representation, because of their complex and changing shape in time. Nevertheless, our method is robust enough to give them a reasonable form.

4.5 Conclusion

A novel dense multi-view stereo reconstruction pipeline has been presented. The whole method is designed to handle the reconstruction of large-scale cluttered scenes taken under uncontrolled imaging conditions, a scenario where traditional multi-

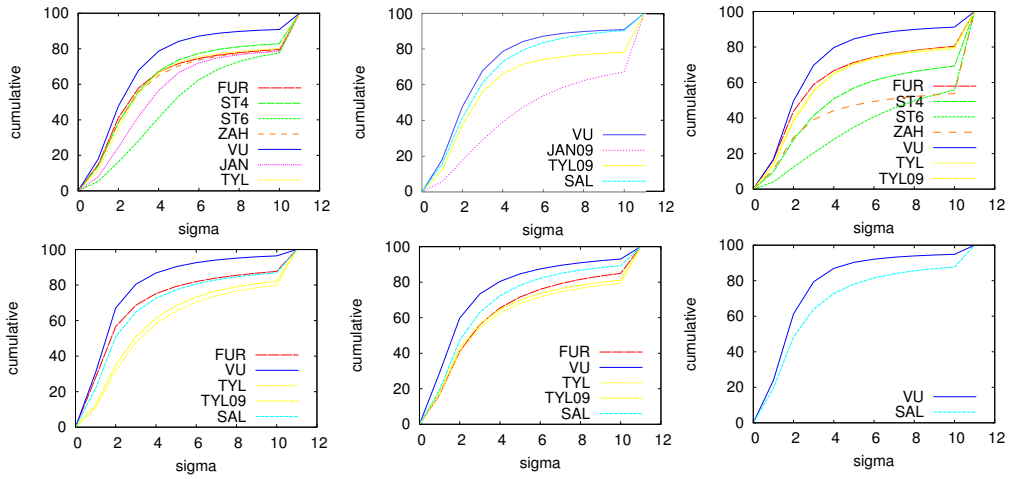


Figure 4.8: ***Relative error cumulated histograms.*** From left to right, up to down, the relative error cumulated histograms respectively for the *fountain-P11* (2 first histograms), *Herz-Jesu-P8*, *entry-P10*, *castle-P19*,*Herz-Jesu-P25* data set. Legend is the following: FUR for [Furukawa and Ponce, 2008], ST4 for [Strecha et al., 2004], ST6 for [Strecha et al., 2006], ZAH for [Zaharescu et al., 2007], TYL for [Tylecek and Sara, 2009], TYL09 for [Tylecek and Sara, 2010], JAN for [Jancosek and Pajdla, 2009], JAN09 for [Jancosek et al., 2009], SAL for [Salman and Yvinec, 2009] and VU for our work. On all data sets, the measurements clearly confirm our better results, both in accuracy and completeness.

view stereo methods are either not applicable or have completeness and accuracy issues in part due to a lack of a correct treatment of visibility issues. The initial surface reconstruction problem is cast as the recovery of a visibility-consistent surface from the Delaunay triangulation of a quasi-dense point generated from the input images. This problem is reduced to a binary labeling of tetrahedron that can efficiently be computed with a minimum s-t cut: the obtained surface is both complete and close to the ground truth and serves as a coarse initial estimate of the scene or object of interest. Its accuracy is then improved by a carefully designed and scalable variational refinement. The full multi-view stereo pipeline has been demonstrated on a number of large-scale scenes. Its results are visually and quantitatively more accurate and complete than state-of-the-art techniques. This pipeline will be adapted for even larger data-sets in the rest of this thesis.

The variational method of the pipeline was combined with geometric primitives (planes, spheres, cylinders, cones and tori) in a hybrid multi-view stereo [Lafarge et al., 2010]. The obtained results have smaller size and still preserve the quality of surface with the primitives that are suitable for urban scenes.

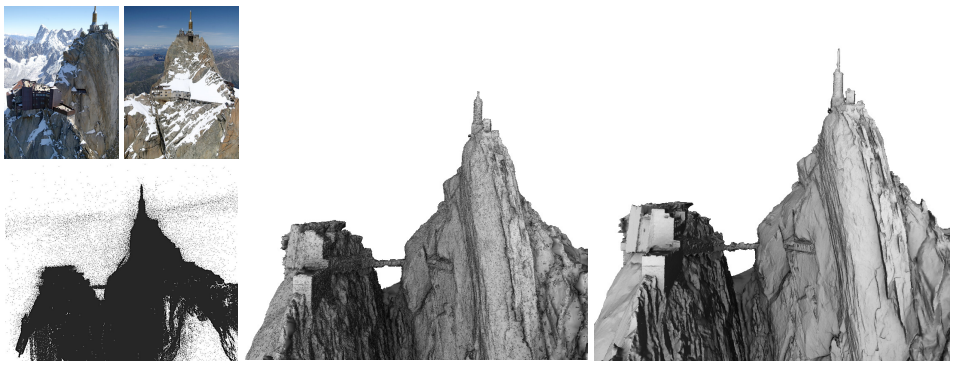


Figure 4.9: ***Results on Aiguille-du-Midi data set.*** From left to right: two sample images taken from a helicopter (© B.Vallet/IMAGINE), point cloud from interest points, initial surface and our final reconstruction.

Figure 4.10: ***Refined mesh on ground-level scene Calvary.***

Figure 4.11: ***Results on Cluny data-set.*** Cluny Abbey built from 30 images taken from a balloon (© B.Vallet/IMAGINE) and an image of data-set.

Figure 4.12: ***Results on Entrevaux data set.*** Top row: final model of Entrevaux with texture. Two bottom rows: non texture mesh taken from 109 images form a helicopter (© IMAGINE/CSTB), seen in 2 views associated with similar images.

Chapter 5

Surface triangular mesh merging

Contents

5.1 Introduction

In 3D acquisition problems, both active (laser range scanning) and passive (multi-view stereo), large scale reconstruction is usually a big issue. Lack of memory, an in-core method, which stores and processes data in the computer memory, is severely penalized. Without changing this in-core method, there is a general paradigm to solve this issue: divide the problem into smaller one, solve each small problem, and combine those partial solutions into the final solution (*Divide and Conquer*). When the problem is the surface reconstruction, which produces a big mesh: instead to obtain the mesh, we will get many separated partial meshes. However, combining these separates meshes is not evident.

In this chapter, we are interested in producing a final mesh from overlapped separate meshes *in common coordinate*. While the union of separate meshes is enough for graphic rendering, or depth map benchmark, it is undesirable for some mesh processing, like smoothness, edge collapse decimation. Such operations will

shrink each mesh. Thus, merging overlapped meshes into a single, topologically correct mesh is essential for mesh processing and simulation.

The problem of mesh merging resides in the overlap of meshes: double walls, and gaps. Thus, it can be solved as a model repair problem. One approach is to consider the union of input meshes as a whole input and try to correct artifacts. Although this approach can fix various artifacts, it might resample input meshes unnecessarily because it breaks topology information of input meshes. Any remeshing, and surface reconstruction algorithm can be used in this approach: distance function introduced by [Curless and Levoy, 1996], Power Crust by [Amenta et al., 2001], Poisson surface by [Kazhdan et al., 2006], implicit surface from polyhedron soup by [Shen et al., 2004], surface meshing by [Boissonnat and Oudot, 2005] and many more.

It is desirable to preserve input meshes as much as possible in mesh merging to economize computing resource. Mesh zippering by [Turk and Levoy, 1994] created connectivity in overlapped areas by adding new vertices and computing geometrical intersection. TransformMesh by [Zaharescu et al., 2007], [Zaharescu et al., 2011], also computed mesh intersection in a different context (mesh deformation rather than mesh merging). These methods require considerable care to handle connectivity in the overlap area to combine meshes. A better approach is resampling the whole overlapped area and preserving the non-overlapped one. Any remeshing algorithms can be applied in the overlapped area to create a single surface, providing that this new surface is appropriately connected with the remaining non-overlapped area. Some methods are proposed in that direction: Poisson surface with boundary by [Huang et al., 2007], local Marching Cubes by [Wojtan et al., 2009], advancing front by [Scheidegger et al., 2005], Laplace surface editing by [Sorkine et al., 2004], SnapPaste by [Sharf et al., 2006]. Likewise, we propose a *novel algorithm*, that remeshes overlapped areas, based on Constrained Delaunay Triangulation, which succeeds in combining many dense meshes into a single manifold surface. To our knowledge, this is the first method which automatically merges hundreds of dense surface triangular meshes.

5.1.1 Contribution

The main contributions of this chapter are:

- Novel remeshing method with boundary information, based on *constrained Delaunay tetrahedralization* (or constrained 3D Delaunay triangulation)and *graph cuts* extraction.
- Non-overlapped areas of input meshes are preserved. It can almost be stored on hard disk during the algorithm, which makes it possible to merge a large data-set.

Thanks to its proper properties, Delaunay triangulation is popular in surface reconstruction (excellent survey by [Cazals and Giesen, 2006]). Extraction of surface from a Delaunay tetrahedralization with graph cuts optimization can be found in [Labatut et al., 2007], [Labatut et al., 2009b], [Wan et al., 2010]. Unlike their work, we use constrained Delaunay tetrahedralization instead of normal Delaunay tetrahedralization, and different graph cuts optimization to target merging problem.

Below, we will describe all steps of our method, which are illustrated and validated through various experiments. Next, we will apply the method in a special case where input meshes are associated with a box partition. This case is used for merging process in chapters 6 and 7. Finally, we will conclude the chapter and discuss its limitation.

5.2 Merging algorithm in general case

The mesh merging is stated roughly as: *An unknown surface S is approximated with n triangular 2-manifold meshes, sharing some overlaps. Build a single 2-manifold mesh that is geometrically and topologically correct with the surface S.*

The problem is ill-posed because the exact surface itself is unknown. Moreover, there are many ways to mesh a single surface. For instance, the unknown surface may be exactly the union of n input meshes, including all overlapped parts (which is unlikely). Normally, we expect that the surface is regular and smooth enough. In applying Constrained Delaunay tetrahedralization (CDT) [Si, 2010] (appendix A.2), our algorithm consists of 3 steps:

1. Detect overlapped part for each input mesh.
2. Consider a *piecewise linear system* (PLS) (appendix A.2) consisting all vertices of overlapped area, and a set of triangle rings F (of non-overlapped area for each mesh) around it. Compute its CDT. Consider a graph dual of the tetrahedralization. Assign edge weights, compute label nodes and extract a surface according to the s-t mincut. Collapse Steiner points.
3. Cut this extracted surface along the edges between triangle rings F and the remaining non-overlapped area with a graph cuts optimization. Combine this surface to remaining non-overlapped area.

We explain each step in the following sub-sections.

5.2.1 Overlap detection

Due to geometrical errors (input meshes are not exactly on the same surface) and combinatorial incoherence (vertices are not identical), overlaps are unlikely intersections of the input meshes. We propose a simple and inexpensive way to detect

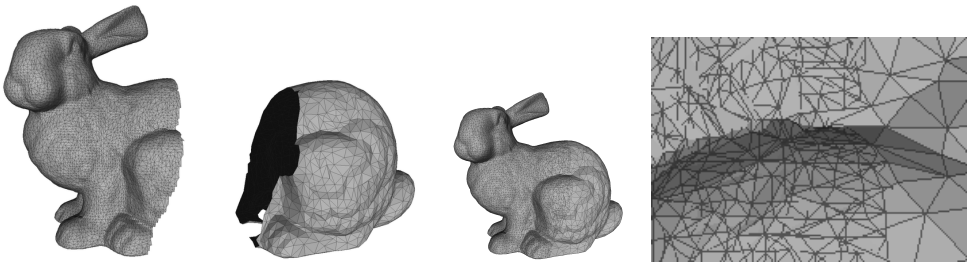


Figure 5.1: The illustration with Stanford's Bunny: left to right: 2 meshes of different resolution, their union in the same coordinate, a zoom in overlap. Note overlapped area, although faithful to the surface, is totally inconsistent.

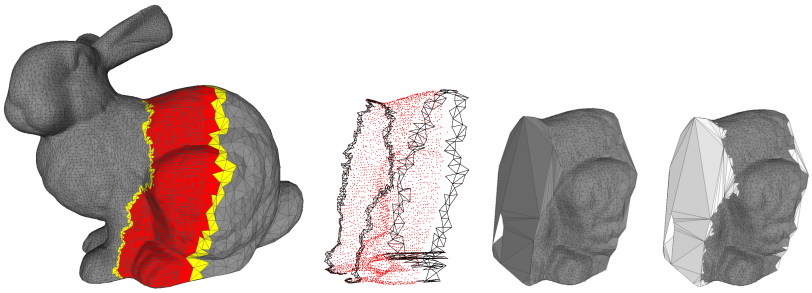


Figure 5.2: From left to right: overlap detection (overlap: red area, around triangles: yellow area, the remaining non-overlap: grey area), PLS of overlap points and constrained triangles, extracted surface from tetrahedralization, a junction surface cut(dark grey area).

overlaps. It would be safe to suppose that the distance of an input mesh to its ideal surface is much smaller than its edge length (there is no point building a mesh whose resolution is smaller than its estimated error). We associate each facet ABC with a sphere $B(G, r)$ which entirely containing it, where G is its mass center, $r = k \times \max\{GA, GB, GC\}$ with $k > 1$ ($k = 1.1$ for our experiment). We detect *the intersection of spheres of different meshes*. We use a kd-tree for spheres (appendix A.5) to speed up the collision detection: spheres in different leaves are disjoint, then intersections occur only in leaves. These intersected spheres correspond to overlapped facets. We collect overlapped facets and rings of non-overlapped facets around (in practice, for each mesh, we take 2 layers of non-overlapped facets: one for PLS in the next step, one for fixing 2-manifold in the merging step). The remaining non-overlapped triangles can be saved on hard disk to economize memory, because we will only use it at the end, simply as an unmodified part of the merged surface.

5.2.2 Graph cuts on Constrained Delaunay Tetrahedralization

We consider collected facets from the previous step. To ensure the numerical stability, we determine vertices which have the same position(or very close together).

For this purpose, we detect collision of tiny spheres centered in every vertex, with a radius of $\varepsilon/2$ (this means their distance is less than ε). We set $\varepsilon = 10^{-6}$ in our experiments. We re-use a kd-tree to accelerate the collision detection.

Next, let P be the set of all vertices of overlapped facets. For each vertex, we compute its normal vector, as a weighted sum of normal vectors of facets around it. For each input mesh, consider a ring of non-overlapped facets, which share a vertex with an overlapped triangle. F denotes the union of these rings. The overlap detection ensures that no triangles in the non-overlapped areas of different meshes can intersect. However, facets in the non-overlapped area of the same mesh may intersect (due to input error). For each input mesh, we detect and remove intersected triangles (with a kd-tree of triangles for acceleration) of the set F. We need remove these self-intersected triangles to ensure F is a PLS. We denote by E the set of edges which separate F with other non-overlapped facets. Considering a PLS which consists of the point set P and triangle set F as a constraint, we compute the CDT by Si Hang's software *tetgen* [Si, 2009] [1] (Fig. 5.2). As a result, we obtain a CDT with possible Steiner points inserted on the edges of constrained triangles. We register all Steiner points with the edges containing it (by using a kd-tree for segments, appendix A.5). Hence, we will know an input facet $f \in F$ is preserved or fragmented into smaller triangles in the tetrahedralization. An infinite vertex is added in the CDT which joins all facets of its convex hull, so that an open surface can be extracted (as [Labatut et al., 2007], [Labatut et al., 2009b]).

We consider a dual graph of this triangulation: nodes correspond to tetrahedra, edges correspond to the triangular facets between adjacent tetrahedra. Each node is linked to the source s and the sink t. We determine a cut C minimizing an energy: $E(C) = E_{data}(C) + E_{quality}(C)$, where each term corresponds to a cost in the graph cuts framework:

$$E_{data}(C) = \sum_{v_i \in S \setminus \{s\}} t_i + \sum_{v_j \in T \setminus \{t\}} s_j, E_{quality}(C) = \sum_{\substack{v_i \in S \setminus \{s\} \\ v_j \in T \setminus \{t\}}} w_{ij}. \tag{5.1}$$

The set S (source $s \in S$) and the set T (sink $t \in T$) contain tetrahedra respectively in front of and behind the surface. Hence, if we favor a facet i to be in front of, or behind the surface, we raise s_i or t_i respectively. These values are computed based on constrained facets and the vertex normal vectors in the tetrahedralization:

- Intialization: set all values s_i, t_i, w_{ij} to 0.
- *Constrained costs* : for all constrained (sub)triangles f of F, which belongs to

[1] `http://tetgen.berlios.de/`

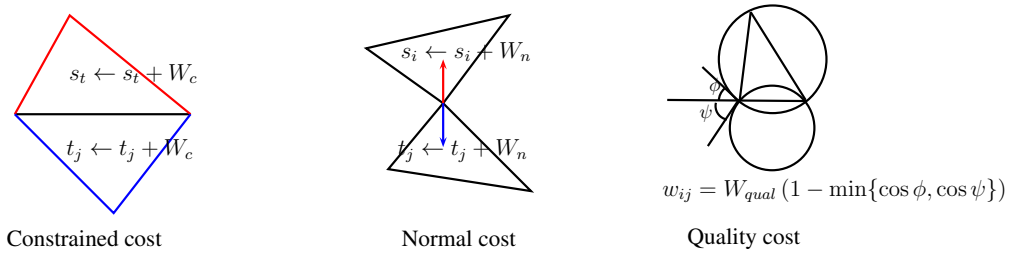


Figure 5.3: Graph cuts costs. A triangle represents a tetrahedron. Red (blue) triangles/normal vectors represent the direction outwards (inwards) the surface.

2 tetrahedra: v_i (in front of the facet) and v_j (behind the facet): $s_i \leftarrow s_i + W_c$, $t_j \leftarrow t_j + W_c$ (W_c is a big value). These constrained costs force constrained facets with right orientation appear on the final surface.

- *Normal costs*: for every vertex V with its normal vector: $\vec{N}$ (which is oriented outside the surface). If there is a finite tetrahedron i incident to V, and containing $\vec{N}$, we set: $s_i \leftarrow s_i + W_n$ ($W_n > 0$). If not: $s_i \leftarrow s_i + W_n$ (for all infinite tetrahedra i incident to V). We apply the same for the opposite normal vector $-\vec{N}$ and t_j: $t_j \leftarrow t_j + W_n$. The normal costs encourage the surface to pass the soup of facets in overlapped areas.
- *Quality costs*: we use the quality cost by [Labatut et al., 2009b], which penalizes facets unlikely to appear on a densely sampled surface. Concretely, with a facet f incident to tetrahedra v_i and v_j: $w_{ij} = W_{qual} \times (1 - \min\{\cos(\phi), \cos(\psi)\})$, where $W_{qual} > 0$, ϕ and ψ are the angles of the facet with the cumscribing spheres of tetrahedra v_i and v_j. This term favors a regular surface.

After the optimization, we extract a cut C, and the associated surface S_1. Because of the extreme value W_c, all constrained triangles will appear on the surface S_1 (with probably some Steiner points). We remove Steiner points as follow: for a Steiner point D inside a constrained edge AB, with $DA \leq DB$ we replace D with A, and we remove all degenerated triangles. It is like a continuous move (without self-crossing) of Steiner points towards triangle vertices: the union of sub-triangles inside a constrained triangle ABC remains ABC during the move. Finally, while all Steiner points are identical with A, B, C, one sub-triangle becomes the constrained triangle ABC (and the others are degenerated and removed). We denote by S_2 this obtained interpolated surface, which keep the constrained facets of F.

5.2.3 Graph cuts on the extracted surface

In the step described above, we marked E as the set of edges separating F from the remaining non-overlap parts. Now, we need to cut the interpolated surface S_2 by

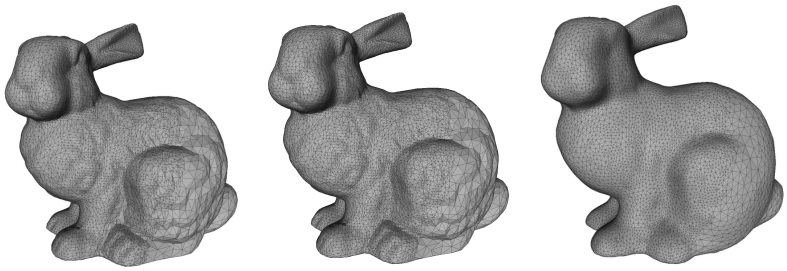


Figure 5.4: From left to right: 2 input meshes in same coordinate, final merged mesh, and its smooth version (applied Laplacian smooth operation).

edges in E, that preserves all triangles in F and triangles in overlapped areas(see Fig. 5.2).

We apply a graph cuts optimization again: consider a graph in which nodes are facets of S_2, edges correspond to sharing edges of 2 adjacent facets, with the cost as the edge's length. We mark all triangles of F in S_2 as 'inside' (they link to the source s with very high cost), all other triangles with an edge in E as 'outside' (they link to the sink t with very high cost). The optimal cut will be the shortest cut that separates these 'inside' and 'outside' triangles, which passes through all edges of E. Let S_3 be the surface of computed 'inside' facets. S_3 can naturally merge with the remaining non-overlapped triangles from the first step, because they share same border edges. Nevertheless, this merged mesh may not be 2-manifold. Indeed, while S_3 is the surface separating different label tetrahedra, it can contain some edges incident to more than 2 triangles, or some vertices incident to non-connected facets. To correct manifold issue, we consider another ring of non-overlapped triangles around S_3, add some identical vertices to make all vertices of S_3 is 2-manifold, before merging with the remaining non-overlapped area. So if input meshes are 2-manifold, the final merged mesh is surely 2-manifold, which is topologically correct, suitable for further processing (see the Laplacian smoothness in Fig. 5.4 and Fig. 5.5).

5.2.4 Experiments

Beside using *tetgen* for CDT, we use the CGAL library for geometry processing, Boost Graph Library implementation of Kolmogorov for graph cuts optimization([Boykov and Kolmogorov, 2004]). We set the following cost parameters for all experiments: $W_c = 10^9$, $W_n = W_{qual} = 1$. At first, we tested with 2 classical Stanford data-sets: Bunny and Armadillo. We cut each mesh manually into small meshes with overlaps. We applied a different decimation for each sub-mesh so that they can not longer "stick" together. Our merging method gave a geometrically and topologically correct final mesh (Fig. 5.4, 5.5).

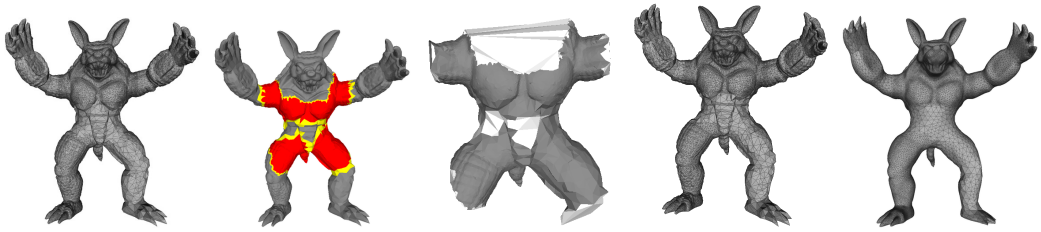


Figure 5.5: From left to right: 6 input separate overlapped meshes, overlap detection (overlap:red, constrained triangles:yellow, remaining non-overlap: grey), junction surface (grey), merged mesh, smooth merged mesh.

We also tested our method to merge input partial surfaces reconstructed from multi-view stereo. We considered two data-sets of city scenes, consisting of 4 and 347 meshes. Each mesh of the scene is built separately. These data-sets are challenging, not only because of their size, but also because they contain noise from camera calibration and multi-view reconstruction. As overlaps are always inside intersection of the bounding box of meshes, we use this information to discard facets outside this intersection zone and to accelerate overlap detection. Fig. 5.6 shows a 4-mesh data-set in different coordinates, an overlap zone (triangles discarded by bounding box do not appear) and interpolated surface, merged mesh and its decimated version (quadratic collapse edge decimation by Meshlab with 5% of total merging facets). The correctness of the decimated surface in the overlapped area shows the merging process is successful. Fig. 5.7 compares an overlapped area before and after the merge: the topology and small gaps are corrected. For the City-347 data-set, we show in Fig. 5.8 the overlapped zone, the interpolated surface, and the decimated merged surface. Although the overlap is quite complicated due to hundreds of inputs, a junction surface has been appropriately extracted (black part in upper right figure in Fig. 5.8).

However, several wrong facets have been observed in the interpolated surface, which connects far vertices in different overlap zones. Fortunately, they are rare and can be easily discarded by setting a threshold of triangle size after extracting the interpolated surface, or setting a strong penalty for big triangles in graph cuts optimization. This result show that our approach is robust and efficient to handle large data-sets.

The following table shows information about data-sets and running time (I/O included), in a laptop equipped with 4 GB memory, a 2GHz processor Duo Core (without parallelization), and a Solid State Driver (SSD):

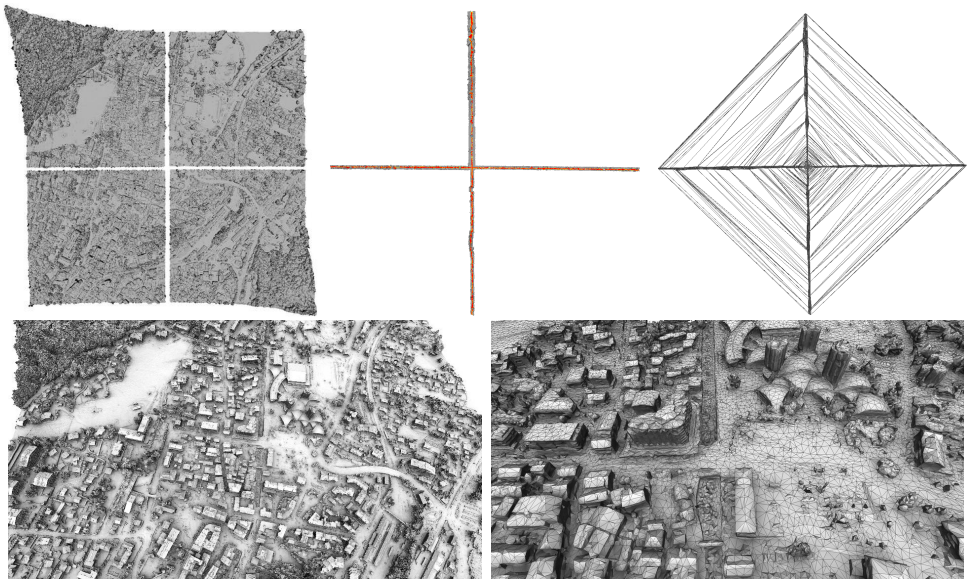


Figure 5.6: Merging of 4 meshes. First row: 4 separate meshes in different coordinates, overlapped zone (red: overlap facets, yellow: constrained facets, gray: remained non-overlapped facets), interpolated surface S_2 (black:"inside", white: "outside"). Second row: merged surface of $15M$ facets, zoom on a decimation of merged surface with 5% of the total facets.

Data-set	# input meshes	# PLS points/facets	# final facets	seconds
Bunny	2	3921/613	24K	0.4
Armadillo	6	9984/1250	53K	1.1
City-4	4	105556/38976	15M	81
City-347	347	961813/384676	43M	323

5.3 Merging meshes from partition of bounding box

In this section, we handle a particular merging problem which usually occurs in practice: merging meshes from a partition of bounding boxes. We suppose a 3D

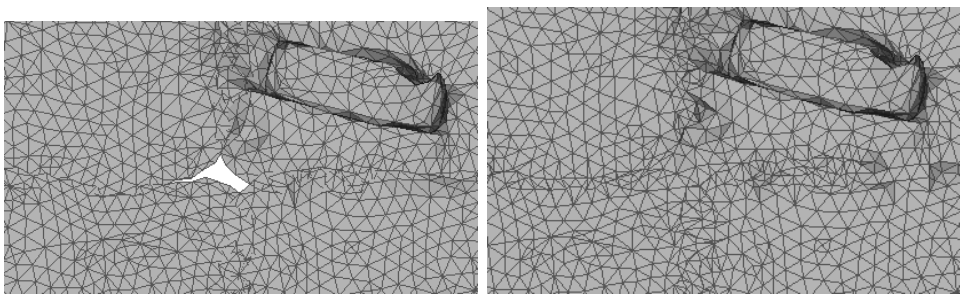


Figure 5.7: Zone containing overlap of 4 separate meshes: union of meshes and merged result. Note small gaps are completed and overlap is corrected.

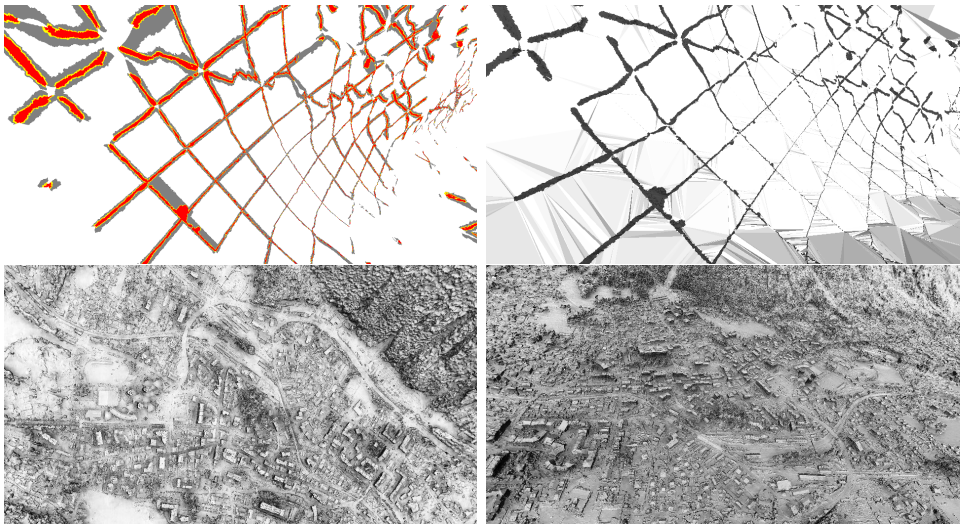


Figure 5.8: Merging of 347 meshes. First row: overlapped zone (red: overlap, yellow: constrained facets, grey: others non-overlapped facets), interpolated surface S_2(black: "inside", white: "outside"). Second row: decimated merged surface with 25% of total $43M$ merged facets.

object/scene is too large to reconstruct directly (by some active/passive 3D reconstruction methods). Its estimated bounding box is then partitioned into small bounding boxes, such that we can reconstruct each piece of the surface for each small box. The input meshes slightly exceed its associated box to limit the border effect in the reconstruction.

While this might be solved as the general method in 5.2, we had better take advantage of the partition of the bounding box. Firstly, for each box b (in the bounding box partition) and its associated mesh M, we do not need to keep any vertex of M outside b in the final merged mesh. Effectively, we expect the part of M outside b is better reconstructed by some other meshes coming from other boxes. Secondly, we should keep as much as possible the part of M inside b in the final mesh, because no other mesh would represent the scene better than M inside the box b.

One straightforward strategy to respect the criteria is:

1. For each box b and its associated mesh M: we define a facet of M as 'inside', 'outside' if it is completely inside, outside the box b (all its 3 vertices are inside, outside the box b respectively). We define 'wall-cut' facets of M the other facets. (at least one vertex inside b and one outside b).

2. We consider a ring of 'inside' facets of M which share at least one vertex with 'wall-cut' ones: they form a set of constrained facets in the CDT.

3. We remove all 'outside' and 'wall-cut' facets. We run CDT for all constrained facets of every input mesh M and we reuse the rest of the general merging algorithm.

We consider in particular the case when input meshes share a common set of vertices. It may happen when the surface is reconstructed from a point cloud in each box and the method keeps input points as vertices of the surface (*e.g.* merging visibility-consistent meshes in chapter 6). There are 'wall-cut' facets $A_1A_2A_3$, where for $i = 0, 1, 2$: A_i is inside a box b_i, and A_i belongs to the associated input mesh M_i of b_i. We should let such facets appear in the final mesh, thus we do not remove them but set them as constrained facets.

We illustrate the merging strategy with Poisson surface reconstruction reconstruction methods of Stanford bunny data-set. We align its range images to obtain a point cloud of 362 K points with an estimated normal at each point. We partition the bounding box of the point cloud in 5 boxes, in which each contains roughly 1/5 the number of points (Fig. 5.9. For each box, we consider all points inside and 20% more points around the box to obtain 5 point sets. We run a Poisson surface reconstruction method for each set (Fig. 5.10). In the end, we merge 5 partial meshes to have the final mesh (Fig. 5.11).

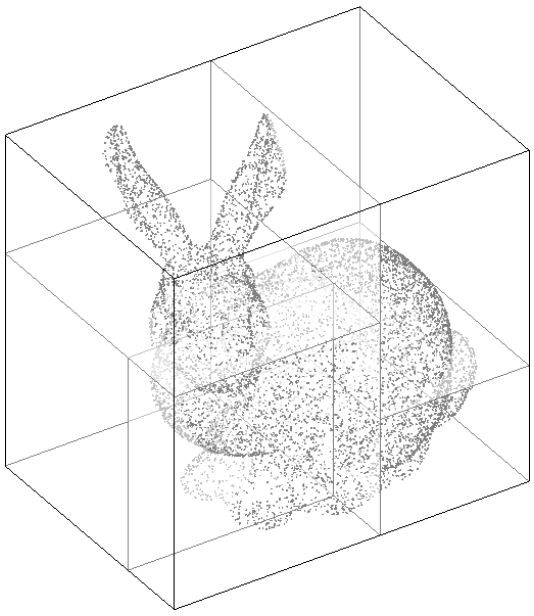


Figure 5.9: Partition of point cloud of Standford Bunny in 5 partitions. Each partition has almost the same number of points.

We note that input partial meshes do not match quite well: there are visible seams in the merged mesh (Fig. 5.11). However, the algorithm is quite successful to recover the mesh topology. Still, there are few small holes in the region around

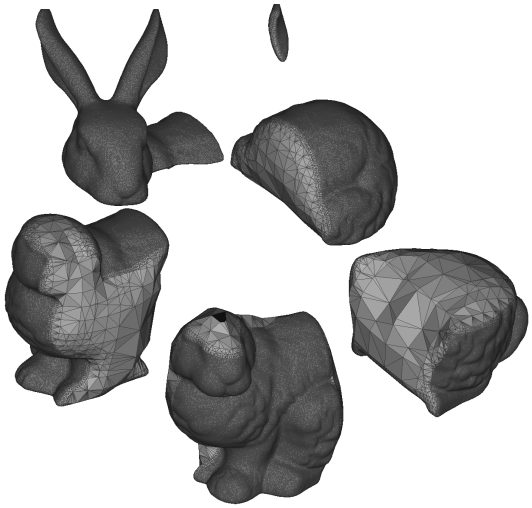


Figure 5.10: Partial meshes from Poisson surface reconstruction (in different coordinates system for better visualization).

the seams that the method does not recover.

5.4 Conclusion

We proposed a fully automatic algorithm to merge many meshes with overlaps, based on CDT and graph cuts. Our method remeshes overlapped areas only, while keeping the connection with non-overlapped parts thanks to CDT. The computation resource was focused on extraction of the junction surface and graph cuts optimization used in tetrahedralization and surface. The experiments show the method's high effectiveness and performance in merging many dense triangular meshes. This merging method does not change vertices position, therefore, does not correct any geometrical error. If it is desirable, a variational method can be used as a post process to reduce the seam and the geometry error in the overlapped areas. This merging method will be used for next chapters, when we will handle large scale data-sets. It would be also useful for merging separated meshes, found in other multi-view stereo or view clustering methods such as [Jancosek et al., 2009], [Furukawa et al., 2010].

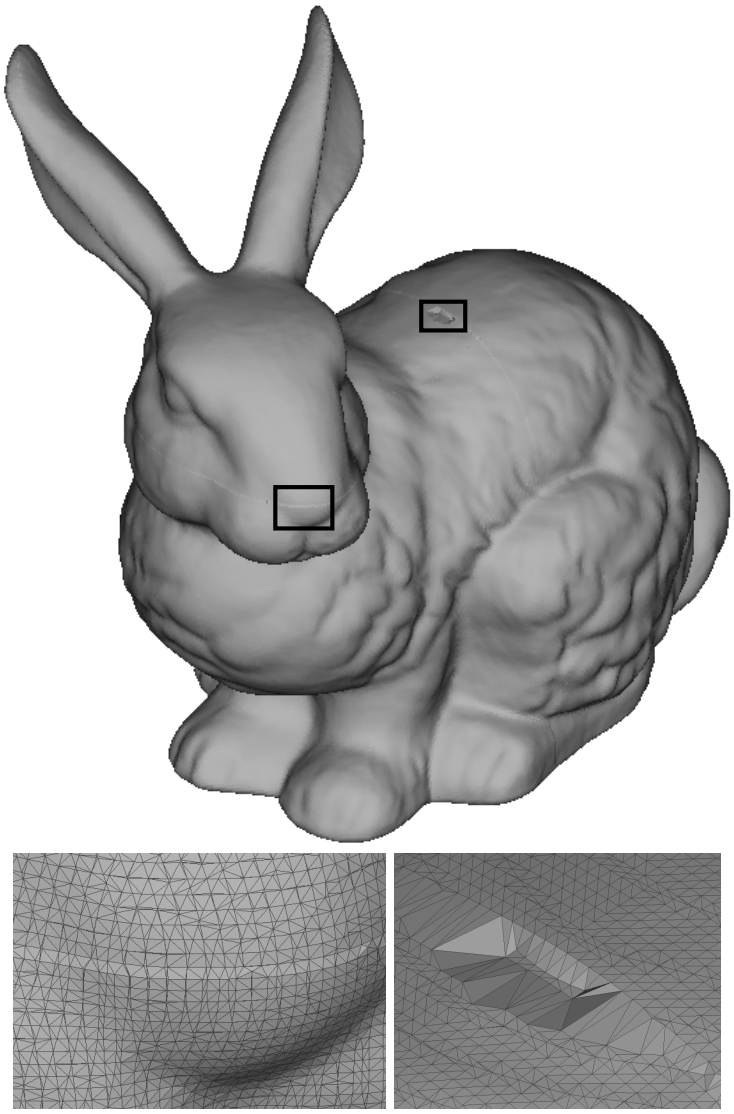


Figure 5.11: Merged mesh from partial Poisson meshes with zoomed details.

Chapter 6

Large-scale visibility-consistent surface reconstruction

Contents

In this chapter, we will study the problem of finding a surface over a large point set associated with line-of-sights. The globally visibility-consistent surface reconstruction presented in chapter 4 might not be applied for large input because it requires large memory in Delaunay triangulation and graph cuts optimization. We will describe how to adapt this method in a convenient Divide-and-Conquer paradigm that tries to conserve the surface quality of the original method.

The contribution of this chapter is as follows:

- Multi-level representation of a point set, from coarse to fine levels. Fast way to compute any point set a representation from a given point set. Equal partition of a point set using a convenient kd-tree.
- Coarse-to-fine extraction of visibility-consistent surface, which gradually removes outliers in each level.

- Adaptation of the original surface reconstruction method for a subset of points, which takes into account the exterior line-of-sights. Test of the mesh merging method presented in chapter 5 in various data-sets.

6.1 Introduction

6.1.1 Motivation

The surface reconstruction in our multi-view pipeline in chapter 4, is a bridge which transforms a discrete 3D model (a point cloud) to a continuous surface (a watertight mesh). According to our experiments, this step is quite fast, comparing to the time dedicated for the remaining steps (plan-sweeping generations of points and the photometric deformation of mesh). Nevertheless, this method requires an in-core 3D Delaunay triangulation as well as an in-core graph-cuts optimization which consumes a considerable amount of memory. Solving this challenge is a key towards a large-scale multi-view stereo.

The problem is stated as: *Given a point cloud* $\mathcal{P} = \{P_0, \ldots, P_{n-1}\}$. *Every point* P_i *remembers 2 or more cameras (associated with a confidence measure) from which it has been triangulated (we called such point a* ***track****). The point cloud has a significant amount of outliers due to mismatches. Extract a triangular mesh* $\mathcal{M}$ *from the point set, which respects as much as possible the visibility constraint of line-of-sights in* $\mathcal{P}$. Its size n is too big for an in-core 3D Delaunay triangulation and a graph cuts optimization of previous globally visibility-consistent surface reconstruction method.

In which follows, we make a brief survey about how the visibility issue has been treated in the multi-view stereo and the surface reconstruction literature.

6.1.2 Work in multi-view stereo with the visibility issue

The visibility (or occlusion) is an compelling issue in multi-view stereo. A camera can only capture the part of the surface which is in its view and is not hidden by other objects or the surface itself. A part of a surface is visible for some cameras but may not be visible for others. This makes point matching more difficult and prone to errors: we can not match a feature of an image to another one that does not see it. Visibility information is then important for the matching process which is a key to reconstruct a surface. However, the exact visibility information is only available if the surface is known.

Fortunately, it does not mean the visibility is an intractable problem. Two features with good matching cost will have a high chance to correspond to a 3D point of the scene. That is why visibility (or occlusion) can be considered as outliers during

the matching process. Most depth map based multi-view methods simply ignore visibility in their first matching step, such as [Kolmogorov and Zabih, 2002],[Strecha et al., 2003], [Strecha et al., 2004], [Gargallo and Sturm, 2005], [Strecha et al., 2006], [Goesele et al., 2006], [Goesele et al., 2007], [Furukawa and Ponce, 2008]. Other methods handle visibility by iteratively updating depth maps and visibility information in a Bayesian framework: [Strecha et al., 2004], [Gargallo and Sturm, 2005],[Tylecek and Sara, 2009]. [Sun et al., 2005] did not use probabilistic but still iteratively optimized occlusion and disparity maps. [Broadhurst et al., 2001] carved a volume with the visibility probability of voxels, [Hernández et al., 2007] used an evidence of visibility as a balloon force in a graph cuts optimization framework.

The nature of self-dependency between the surface and the occlusion leads to variational methods handling visibility from an initial estimation of surface. [Faugeras and Keriven, 1998], [Hernández and Schmitt, 2004], [Pons et al., 2007] used the occlusion implicitly in each optimization iteration by z-culling. [Delaunoy et al., 2008] even included an occlusion formula in its gradient computation.

Although occlusion is a problem for image matching, it does help the estimation of the surface. A point on the surface is seen by a camera only if there is no obstacle between the camera and the point (we ignore transparency, reflectance). It is called 'empty space' property of a line-of-sight. The reconstructed surface should respect this property: a line from a point of the surface to cameras from which it is originated should not cut the surface. This property was exploited in a labeling framework to estimate discrete levels of depth map of a single camera in [Kolmogorov and Zabih, 2001], [Kolmogorov and Zabih, 2002]. However, these methods are only suitable for small data-sets with a few number of depth map layers.

In large scale or real-time multi-view stereo where performance is a constant issue, the visibility is usually ignored in the first step. Many methods try to quickly generate a 3D model representation (*e.g.* . point cloud, depth map, patches), followed by a step of visibility filter. [Merrell et al., 2007] generated depth maps by plane-sweeping stereo with GPU, and combined depth maps with visibility constraints. [Furukawa and Ponce, 2008] used visibility information to remove outlier matching patches. However, these approaches handled visibility in a heuristic manner and did not use a global optimization.

There are some recent online algorithms to extract the surface, based on Delaunay triangulation and visibility constrain. [Pan et al., 2009] produced a surface over feature points in real-time with only a webcam. [Lovi, 2010] independently solved a similar problem with different algorithms. While they used Delaunay triangulation and line-of-sights constraint as us, their point clouds, resulted from SfM, are much cleaner and sparser than our point clouds. More recently, [Jancosek and Pa-

jdla, 2011] modified the range-data global visibility-consistent surface algorithm in [Labatut et al., 2009b], in order to better construct the surface having low texture.

Our global visibility surface algorithm (chapter 4 and some previous work of Labatut *et al.*) extracts a surface from a Delaunay triangulation of a noisy point cloud. The goal of the method is two-fold: (i) remove outliers of the point cloud based on visibility constraint and (ii) extract a globally visibility-consistent surface from the remained clean points, that respects the visibility constraint. Online methods [Pan et al., 2009], [Lovi, 2010] could effectively compute a surface that respects the visibility. However, we think it would be difficult to remove a large amount of outliers from a noisy point set based on their heuristic approaches. Effectively, the point sets resulted from SfM in [Pan et al., 2009] were free of erroneous mismatches, so that they could assume a Gaussian model of outliers around the real surface. In our point sets, the outliers are very far from surface and can form ghost structures, which are impossible to model them. [Lovi, 2010] which used incremental space-carving of Delaunay triangulation with line-of-sights, also handled relatively clean SfM point set.

6.1.3 Work on large scale surface reconstruction

Surface reconstruction (or surface fitting) from a point set is a broad field in computer vision that we do not intend to make a complete and thoughtful overview in this thesis. We mention only general tendencies, specially ones that can handle large point set.

The problem is stated as computing a surface $\mathcal{S}$ from a set of points $\mathcal{P} \subset \mathbb{R}^3$, which is the result of a 3D acquisition technique. The surface $\mathcal{S}$ should match the surface of the original model both geometrically and topologically. Depending on the surface and the sampled points, computation of the surface is quite challenging because of noise, outliers, and varying density.

Various methods have been proposed to solve this problem. Delaunay-based methods compute the surface as a subset of facets in a Delaunay triangulation (or its variations): the first proposed method by [Boissonnat, 1984], Crust by [Amenta et al., 1998], Cocone and variations by [Amenta et al., 2002] [Dey and Goswami, 2006]. Review and analysis of Delaunay methods can be found in an excellent survey of [Cazals and Giesen, 2006].

[Bernardini et al., 1999] proposed a Ball-pivoting algorithm to build a mesh from the point set without computing the triangulation. Other methods define the surface as a level-set of a function that represents the input point set. The function can be distance functions [Hoppe et al., 1992], distance functions with visibility constraint [Curless and Levoy, 1996]. Another choice is using the indicator function,

which has the value 1 inside the object and 0 outside. Solving a Poisson equation related to the indicator function, [Kazhdan et al., 2006] designed an efficient method, which is robust to sample density and outliers. The Poisson surface reconstruction has been widely used to produce a water-tight mesh from a point set in recent multi-view stereo methods [Furukawa and Ponce, 2008], [Tylecek and Sara, 2010].

Those above methods usually handle range scale data, up to hundred million of points (The Digital Michelangelo Project [1] . Many were designed to run in an out-of-core or streaming manner.

In general, all of them can be used within a divide and conquer approach. For example, by dividing the domain space into blocks, the range-image volumetric merging in [Curless and Levoy, 1996] reconstructed pieces of surface independently that were stitched together by identifying and merging common vertices between neighboring blocks [Levoy et al., 2000]. Advancing-front algorithm as [Bernardini et al., 1999] was easily implemented in out-of-core extension. Several other streaming methods were proposed in order to reconstruct the surface gradually by keeping small input data in-core [Pajarola, 2005], [Bolitho et al., 2007].

Therefore, large-scale point set is not an issue for surface reconstruction methods. Nevertheless, the point sets usually come from range-scanning is relatively clean. The outliers of point set generated by multi-view stereo can be very arbitrary (a survey in [Labatut et al., 2009b] showed that beside the presented method, only Poisson surface reconstruction can remove a significant number of outliers).

We conclude that there is no other method in both multi-view stereo and surface reconstruction community which can response to our problem. That is why we will apply our current visibility-consistent surface reconstruction method in a convenient Divide and Conquer manner for large-scale point sets.

6.2 Divide and Conquer algorithm

This Divide and Conquer (DC) approach consists in partitioning a set of tracks $\mathcal{P}$ into small subsets so that we could use the original surface reconstruction method for each subset to obtain many partial meshes. At the end, these partial meshes are combined to produce the final result (with a merging algorithm in chapter 3). However, using the original method in each subset directly will produce partial mesh with two flaws. First, while each partial mesh is visibility-consistent within the tracks of its associated subset, it may not respect the visibility constraint from other parts. Second, the visibility constrained in each subset can be too weak to remove its own outliers (thinking of a subset containing all outliers behind the scene).

[1] `http://graphics.stanford.edu/projects/mich/`

To remedy the problems, we realize that the globally visibility-consistent algorithm fulfills 2 different roles: remove outliers and compute a surface. A DC approach should be designed to accomplish these 2 roles. Therefore, we design a separated algorithm for each role which share many parts in common. First, the point cloud filter algorithm will remove outliers based on visibility-constraint. Second, the surface reconstruction algorithm builds a visibility-consistent mesh.

In order to build a surface from a subset of points that respects the 'free space' property of line-of-sights coming from other subsets, we need to take into account the visibility constrains outside the considered subset. When we compute Delaunay triangulation for each subset (as the original method), we need to add the visibility cost of line-of-sights from exterior points, if they cut the triangulation of the current subset.

While this modification could remove outliers between the ground truth and the cameras, it can not remove outliers behind the ground truth. The reason is no line-of-sight penetrates these outliers. In the original algorithm, the graph cuts optimization is designed to favor the tetrahedron behind the each line of sight as 'inside'. Thus, the outliers behind the ground truth will be surrounded by 'inside' tetrahedra and then be removed from the mesh. This mechanism is weakened when we break the input points into small parts: if one part has not significant amount of inliers, it could not remove the outliers which are behind the ground truth. Therefore, we should consider the input points as a whole in some sense to remove outliers behind the scene. For it, we will use a multi-level representation of input points, along with a DC paradigm to filter the outliers. After removing outliers, we will run a visibility-consistent algorithm for each subset to create a partial surface, that will be merged into a final surface.

In the following, we will present necessary components before describing the main two parts of the method: point filter and surface reconstruction.

6.2.1 Multi-level representation of a point set

We want to create multi-level representations of the given a set of track $\mathcal{P}$, each point is associated with line-of-sights towards cameras and confidence measures. For this down-sampling, we use a binary kd-tree space decomposition, similar as [Tobor et al., 2004], with some modification and improvement in the implementation. The main difference is that the bounding box of point set is subdivided recursively into 2 local sub-domains by a hyperplane in the middle of its longest axis (not in the median of point set as in [Tobor et al., 2004]). This process produces a kd-tree where each node corresponds to a cluster of subset points of $\mathcal{P}$. A low resolution point set of $\mathcal{P}$ is computed as a set of barycenters of a clustering of points of $\mathcal{P}$,

based on 2 list of indexes that we will soon explain.

We propose some definitions, which are only used in this subsection. Let $S = \{0, 1, \ldots, n-1\}$ the set of indexes of $\mathcal{P}$. We say *a partition* of a set $S = \{0, 1, \ldots, n-1\}$ is a set of disjoint subset of S, whose union is equal to S. Given 2 partitions $\mathbf{Par}_1$ and $\mathbf{Par}_2$, we say $\mathbf{Par}_1$ is *coarser* than $\mathbf{Par}_2$ if each element of $\mathbf{Par}_1$ is a union of certain elements of $\mathbf{Par}_2$, and note $\mathbf{Par}_1 \prec \mathbf{Par}_2$. We also say $\mathbf{Par}_2$ is *finer* than $\mathbf{Par}_1$. A partition of S: $\mathbf{Par}$=$\{\{i_{00}, i_{01}, \ldots, i_{0s_0}\}, \ldots, \{i_{k0}, i_{k1}, \ldots, i_{ks_k}\}\}$ could be written as a pair of a *background permutation* $\{i_{00}, i_{01}, \ldots, i_{0s_0}, \ldots, i_{k0}, i_{k1}, \ldots, i_{ks_k}\}$ of S and a set called *partition mark* $\{0, s_0, s_0+s_1, \ldots, s_0+\ldots+s_k\}$. This representation of a partition as a pair of a back ground permutation and a partition mark is not unique (by mixing the order of subsets in a partition, we obtain other representations). We notice if two partitions $\mathbf{Par}_1$ and $\mathbf{Par}_2$ share a same background and the partition mark of $\mathbf{Par}_1$ is the subset of the partition mark of $\mathbf{Par}_2$ then $\mathbf{Par}_1 \prec \mathbf{Par}_2$. For examples: $n = 6$, $\mathbf{Par}_1 = \{\{0, 4, 5\}, \{1, 2, 3\}\}$, $\mathbf{Par}_2 = \{\{0\}, \{4, 5\}, \{1, 2\}, \{3\}\}$, we have: $\mathbf{Par}_1 \prec \mathbf{Par}_2$ and they share a background permutation: $\{0, 4, 5, 1, 2, 3\}$ with partition marks: $\{0, 3, 6\} \subset \{0, 1, 3, 5, 6\}$.

We will build a permutation p of S and a list of marks m such that for every value $k < n$, we can create a point set of size k from $\mathcal{P}$ (based on p and m) in linear time. In order to create down samples of the point set, we use a kd-tree to partition the point set into clusters. We consider a node of the kd-tree is a pair of index (u, v). With a background permutation $p = \{p_0, \ldots, p_{n-1}\}$ of S, the node (u, v) corresponds to a cluster of all points of $\mathcal{P}$ with indexes: p_u, p_{u+1}, $\ldots$, p_{v-1}. We build the kd-tree as follow: first the root is the node $(0, n)$ corresponding to the whole point set, the background permutation p is simply the identity, the list of mark is $m = (0, n)$. Recursively, for each node (u, v) containing more than 2 points ($v > u+1$), we compute the bounding box of points of this node, and obtain $[x_{min}, x_{max}]\times[y_{min}, y_{max}]\times[z_{min}, z_{max}]$. We suppose without loss of generality that the bounding box has its longest size along x-axis . We cut the bounding box by the hyperplane of the equation $x = x_0 := 0.5(x_{max} + x_{min})$. Each point P_i whose index $i \in p_u$, p_{u+1}, $\ldots$, p_{v-1} will be put in either the left-child node (when $P_i.x \leq x_0$) or the right-child node ($P_i.x > x_0$). This separation forms a permutation of index of p in the range (u, v): we update the permutation p within this range. Let w the number of nodes in the left-child, then the two child nodes $(u, u+w)$ and $(u+w, v)$ are created. We put element $u + w$ in the list of marks m. A leaf node is a node of type $(u, u + 1)$ because it corresponds to only 1 element of $\mathcal{P}$, which can not be divided.

To arrange the order of the subdivision,we maintain a priority queue Q of nodes that put the node having the longest axis at the beginning. First, we add the tree's root (with its longest axis) in Q. During the tree building, we take out the first

node of Q and create its children nodes which are pushed back into Q. The process terminates when all nodes of Q are leaf nodes.

At the end, we obtain a background permutation p and a list of indexes m: $(m_1, m_2, \ldots, m_{n+1})$, with first element is $m_1 = 0$ and $m_2 = n$. With this information, we can create any multi-level representation of S. To compute a point set of size $k < n$ from S, we extract $k + 1$ first elements of m : $\{m_1, \ldots, m_{k+1}\}$, then we sort them in increasing order to obtain: $\{u_1, u_2, \ldots, u_{k+1}\}$. We then consider k nodes $(u_1, u_2), \ldots, (u_k, u_{k+1})$, that correspond to a partition of set S: $\{\{p_{u_1}, p_{u_1+1}, \ldots, p_{u_2-1}\}, \ldots, \{p_{u_k}, p_{u_k+1}, \ldots, p_{u_{k+1}-1}\}\}$. For each subset of this partition, which corresponds to a node (u, v), we compute the barycenter of all points $P_{p_u}, P_{p_{u+1}}, \ldots, P_{p_{v-1}}$ with weights as their confidence measure. The set of the barycenters is the set of k points down-sampling of the initial point set. For example, a size 1 down-sample of $\mathcal{P}$ is the barycenter of the whole point set with weights as their confidence measure, a size n down-sample is $\mathcal{P}$ itself (an example is shown in the Fig. 6.1). We set the line-of-sight of the new point as the union of line-of-sight of all tracks in its cluster. The confidence measure of each line-of-sight from the new point (created from a cluster) to a camera is set as the summary of all confidence measures of all line-of-sights from points of this cluster to this camera. With this mechanism, we just need to compute the background permutation p and the list of marks m *only one time*, to produce multi-level sets of tracks of any size of $\mathcal{P}$

One difference of our reconstruction from [Tobor et al., 2004] is instead of using the hyperplane-cut in the medium of the point set, we use the hyperplane cutting the bounding box into 2 equals bounding box. Each cluster in our reconstruction is more similar and more compact in the size of volume (but not in the number of points), hence the barycenter of each cluster is better represented all points in the cluster. It produces more faithful multi-level representations of the point set.

6.2.2 Partition of a point set in many equal parts

We want to equally partition a large point set $\mathcal{P}$ to small parts by a bounding box partition. Each part contains all points inside a box of the bounding box partition and nearby points coming from neighboring boxes. The reason of adding these exterior points is to avoid border effect in the local surface reconstruction.

We then formulate the problem as follow: given a point set $\mathcal{P}$ of size n, a positive integer number th, we want to partition the point set into m subsets whose size is not greater than th, in using hyperplanes perpendicular to coordinate axis . We want the number of subset m as small as possible.

Each subset has the size not greater than th, then the number of point: $n \leq$

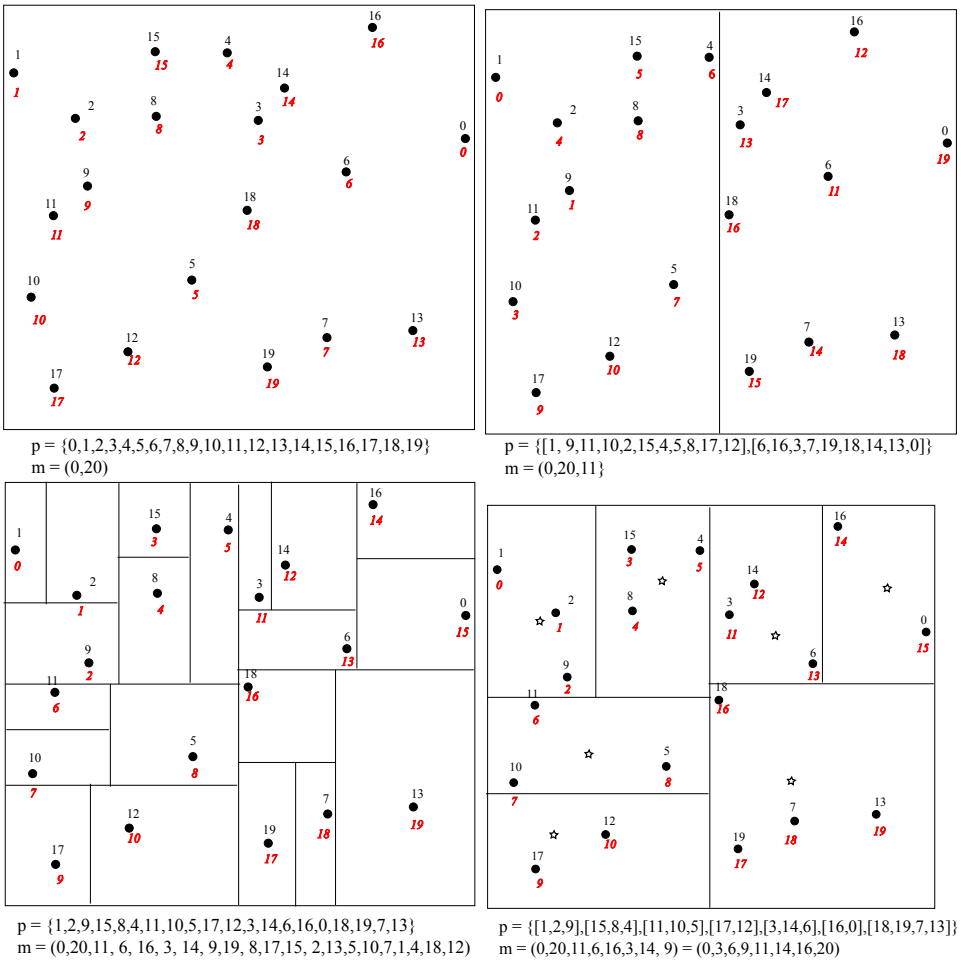


Figure 6.1: Example of multi-level construction of 20 points. The black number above each point is their original indexes. The red number below each point is the index of the permutation p in each step. Below each image we have the background permutation p and the list or marks m. In the first row: the point set and the point set divided by 2. In the second row, left: the point set at the end, right: clustering the point set in 7 points (star point), based on the permutation p and 8 first elements of the list of marks m.

$m \times th$, which leads to: $m \geq n/th$, or $m \geq \lceil n/th \rceil$. It is indeed the best lower bound of m. One solution to obtain that bound is just dividing the point set by hyperplanes of type $x = x_0$, in which each subset has the size of th (possibly except the last one). Nevertheless, such subsets might not ideal to run a local surface reconstruction because its bounding box may be long in one axis, and exceedingly short in another one. While we can not prove that a more 'regular' division leads to a better surface reconstruction, it is intuitive to prefer a regular division.

Therefore, we prefer to cut the point set into $m = \lceil n/th \rceil$ subsets in which each one has a more or less regular bounding box and has almost equal number of points. We propose a recursive solution as follows. Suppose the longest axis of its bounding box is x. If m is even: $m = 2m_0$, we cut the point set by at the medium $x = x_0$

into 2 small sets, so that we will cut later each small set in m_0 parts. If m is odd: $m = 1$: we do not need to cut, otherwise: $m = 2m_0 + 1$: we cut the point set by a hyperplane $x = x_0$ into 2 parts, such that they contain around $m_0/m \times n$ and $(m_0 + 1)/m \times n$ points. The subsets will be cut into m_0 and $m_0 + 1$ subsets. At the end, each subset contains roughly $n/m_0 \leq th$ points. The choice of hyperplanes is done by *the selection algorithm* (finding the first k smallest number in a list), which ensures the linear running time for each step of the algorithm. The total running time is then $O(n \times m)$.

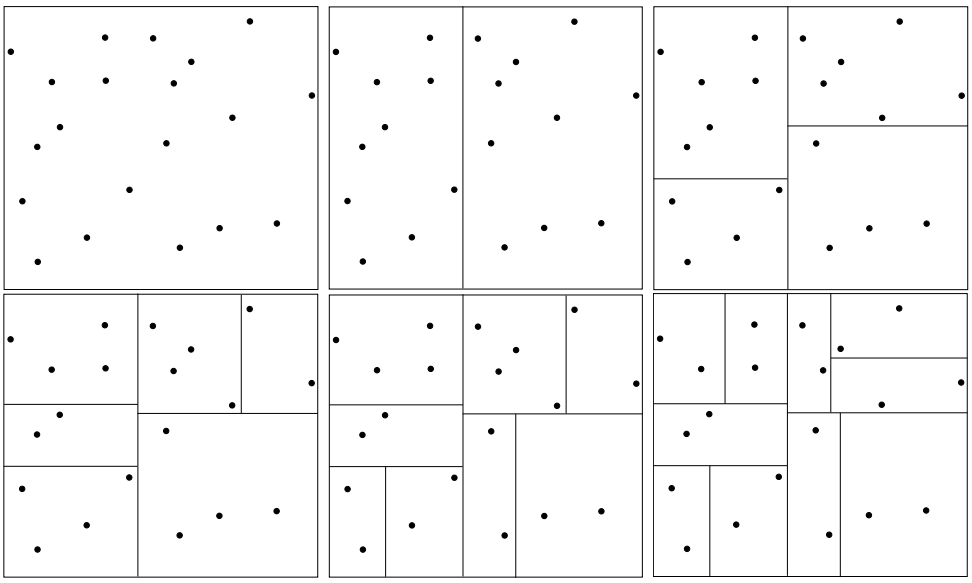


Figure 6.2: Partition of 20 point in 10 subsets. First row, left to right: original point set, division of the point set into 2 equal parts (each will contain 5 subsets), each part is divided into 2 smaller parts with the ratio 2/3 of points number. Second row, the division process continues from left to right until we obtain 10 subsets in the right most image.

We add exterior points in each subset to avoid border effect. We suppose the number of exterior points for each subset, is fixed at th_1, a fraction of the subset's size. By using again a *selection algorithm*, the exterior points are added in linear time $O(n)$ for each subset. Effectively, for each subset, we determine its bounding box. For each point outside the bounding box, we consider its d_∞ distance to the bounding box ($d_\infty(P, Q) = \max\{|P.x - Q.x|, |P.y - Q.y|, |P.z - Q.z|\}$). We select th_1 nearest points as supplement points to the subset. The total running time is again $O(n \times m)$.

6.2.3 Local visibility-consistent surface reconstruction

The goal of this subsection is to adapt the original surface reconstruction method for a subset $\mathcal{P}_k$ of a track set $\mathcal{P}$. We add exterior line-of-sights that cut the convex

hull of the subset in order to remove the outliers in the subset which are between the cameras and the ground truth. Except modified exterior visibility costs (Fig.6.3), the algorithm is the same as the original one.

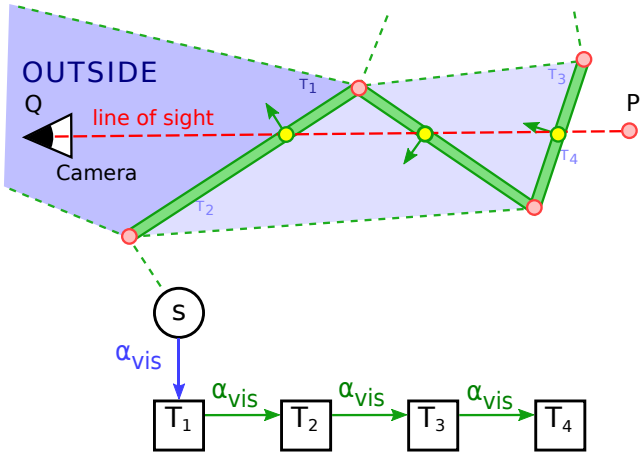


Figure 6.3: Exterior line-of-sight cost.

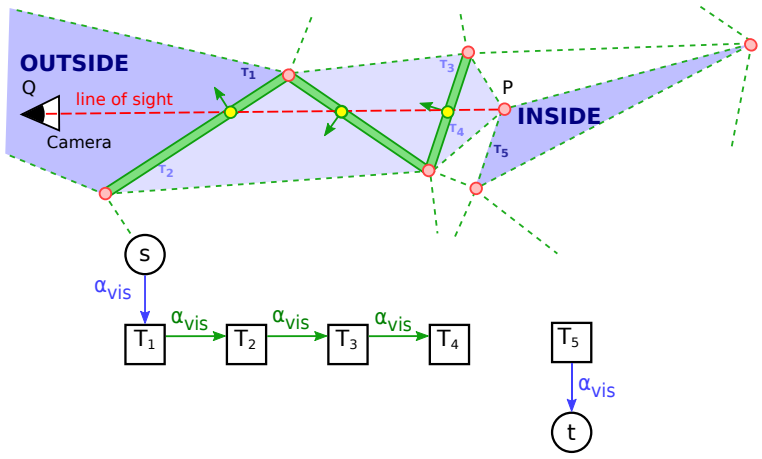


Figure 6.4: Interior line-of-sight cost.

Comparing visibility cost of exterior line-of-sights (Fig.6.3) to visibility cost of interior ones (Fig.6.4), we do not attach the tetrahedron behind an exterior line-of-sight to an 'inside' terminal. This line-of-sight does not belong to the local mesh, we do not need to use this 'inside' cost. Moreover, this cost will favor a minimum cut that cuts this line-of-sight - which is wrong if the subset are all outliers in front of the surface. After assigning cost and applying a graph cuts optimization, we obtain a (local) surface $S = S_k$, with vertex set V.

This local surface extraction will be used in both parts of the DC approach, which are point cloud filter and the general surface extraction.We recall that with the partitioning algorithm in the subsection 6.2.2, the entire point set $\mathcal{P}$ is separated

into many disjoint parts by a partition of the bounding box. In order to limit border effect, each subset $\mathcal{P}_k$, associated with a box B_k, contains all tracks inside B_k and neighboring tracks. In order to remove outliers, we qualify all points in V and inside the box B_k as 'inliers'. Moreover, for each point $P \in V$, we compute the average distance d_P from P to all its neighboring vertices in S. We consider all points Q inside B_k but not in V, which connects to P via the Delaunay triangulation and qualified Q as 'inliers' if $PQ < 0.5 \times d_P$ and $PQ < d_S$, with d_S is the average edge length in S. The reason we add more points is to reduce the number of true negative detection, that will be explained more in the next subsection. If we want to extract a mesh, we consider all facets of S_k having at least one vertex inside B_k.

6.2.4 Multi-level point cloud filter

In this subsection, we describe the first step of our DC method: removal of outliers of the input. We denote by n the size of the input point set $\mathcal{P}$. Given a threshold th, we suppose the surface reconstruction algorithm can run with point set of size no greater than th. If $n \leq th$, we just run the original algorithm for the whole point set and remove all points that do not appear in the output mesh. Unless, we use a multi-level approach as follows.

Firstly, we consider the multi-level representations of a point set (in subsection 6.2.1). We compute the set of tracks $\mathcal{P}^0$ of size th, clustered from the original set of tracks. We run the global visibility-consistent algorithm in $\mathcal{P}^0$ to remove its outliers. Supposing we have already filtered the set of tracks of level k ($k \geq 0$), we will filter the finer representation $\mathcal{P}^{k+1}$. Each outlier p removed from the level k corresponds to a cluster of points in the level $k+1$. Therefore, we remove all tracks of $\mathcal{P}^{k+1}$ corresponding to outliers of $\mathcal{P}^k$. For all remained points in $\mathcal{P}^{k+1}$, we partition it into many equal parts with the threshold value th (subsection 6.2.2) .In each part, we apply the local visibility-consistent surface reconstruction (subsection 6.2.3) to remove outliers of each part. At the end, we obtain the new point cloud as union of point clouds in the level $k+1$. We continue until the final level and obtain a set of clean tracks.

The strong point of this procedure is based on the coarse-to-fine structure of $\mathcal{P}^k$: the result of a coarse level will filter a finer one. We remind that, without this coarse-to-fine reconstruction, only outliers in front of the scene could be deleted. In adopting a multi-level filter approach, we handle the point cloud from the global structure to local ones. The outliers behind the scene are also removed gradually from each level.

Nevertheless, we notice that a cluster point which is removed from a level can not appear in the finer level. A cluster point is built from a set of input points that

may contain both inliers and outliers. A deleted cluster also leads to the removal of its inliers. Therefore, in subsection 6.2.3, we qualify some points near the extracted surface as 'inside', to make these ambiguity points still appear in the finer level. We do not want to add too many points either, because it can add more outliers that are difficult to remove in finer level.

One question is how to choose the number of multi-level set of points and the size of each level. While a big number of levels will make the transition from global structure to local structure algorithms more smoothly, it requires more computation time. On the other hand, too few levels are not enough to remove behind-the-scene outliers. In our implementation, we set $\#\mathcal{P}_0 = th$ and $\mathcal{P}_{k+1} \simeq c \times \#\mathcal{P}_k$ (with c is a constant around 2, depending on the input size), until $1/c$ the size of the original set $\mathcal{P}$. We do not need to run this point cloud filter for the original set, because the final surface reconstruction(presented in the next subsection) also filters outliers.

6.2.5 Partial surface reconstruction and mesh merging

Once the set of tracks is filtered by the first step (subsection 6.2.4) of the DC method, we begin the surface reconstruction. First, we partition the point set $\mathcal{P}$ into many equal parts (subsection 6.2.2), such that each part (its exterior points included) has the size smaller than th. We apply the local visibility-consistent surface reconstruction (subsection 6.2.3) to obtain many partial surfaces S_k associated with the bounding box B_k. At the end, we use the merging method for meshes from a partition of bounding box (see 5.3) to obtain the final surface.

6.3 Experiments

We apply the algorithm for 3 point clouds: Entry-10 (70 K points), Lausanne (3.00 M points) and Chamonix (2.6 M points) (Fig. 6.5). Although the point cloud of Entry-10 is small and not necessary to use the DC algorithm, it is helpful to illustrate and analysis the method. We run all data-sets in a computer with a SSD hard disk, two processor 4 core 2.8 GHz (we use only one core), and 24 GB Memory - the large memory helps run the in-core global reconstruction method, in order to compare with DC approach in term of quality and consumed resource. For all data-sets, we use a various threshold th for local surface reconstruction. ($th = \infty$ for the original method).

The point set of Entry-10 contains 70K points. We use the following thresholds: 1K, 5K, 20K, ∞. We observe in Fig. 6.6 that DC approach is as effective as the original method to remove outliers: there is almost no difference between different thresholds.

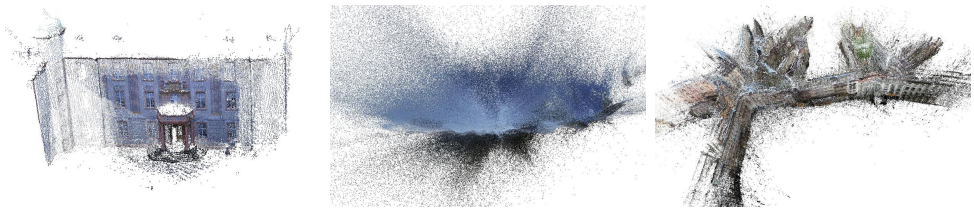


Figure 6.5: Initial noisy point set (with colors) of Entry-10, Chamonix, and Lausanne.

Next, we compare output meshes from different threshold th. In Fig. 6.7, each value th (except $th = \infty$) corresponds to 2 images of the final mesh: left is the union of partial surface inside each box of the bounding box partition, right is the final surface after merging. We can see the effectiveness of the merging method, which combines many small meshes in a singe mesh. There is no different in the facade of the building, where the points are quite dense. The ground varies varies depending on the threshold number th. The running time for Entry-10 is less than 20 s for each threshold (the merging process takes less than 1 second).

In order to understand the impact of multi-level filter process, we compare the reconstructed in case $th = 1K$ with/without multi-level filter in Fig. 6.8. Without the point filter process, the outliers behind the scene are not completely removed.

The point cloud of Chamonix contains 2.6M points, from 2118 images taken from a helicopter. The point set is quite noisy with outliers not only floating in the sky cloud, but also lying under the ground truth (Fig. 6.5). We use the following threshold: $th = 10K, 50K, 200K, \infty$. All values give almost the same point set. Fig. 6.9 compare the input point set and the vertices of the final mesh with $th = 200K$. Most of underground and floating outliers near the surface are removed. Points in the sky above the helicopters exist because there is no visibility constraint to remove them (those points exist because of true or false matching of pixels in the sky).

The final meshes are almost the same in the area where the point set is dense, and again, merging process is quite effective (Fig. 6.10, 6.11, 6.12). In this data-set, if we keep only vertices in the surface during the point cloud filter process, instead of adding some nearby points (subsection 6.2.3), there would be some small details that DC methods do not recover. The lessening of the inliers choice by including points near the surface vertices, limits this unwanted effect.

The point set of Lausanne data-set contains 3.05M points, triangulating from 400 images, visually having fewer outliers than the Chamonix's. We use the same set of thresholds: $th = 10K, 50K, 200K, \infty$. Fig. 6.13 shows the input point against the output vertices with $th = 200K$. Fig. 6.14 shows the final mesh and partial meshes inside each partition with $th = 10K, 200K$. We find almost no difference in

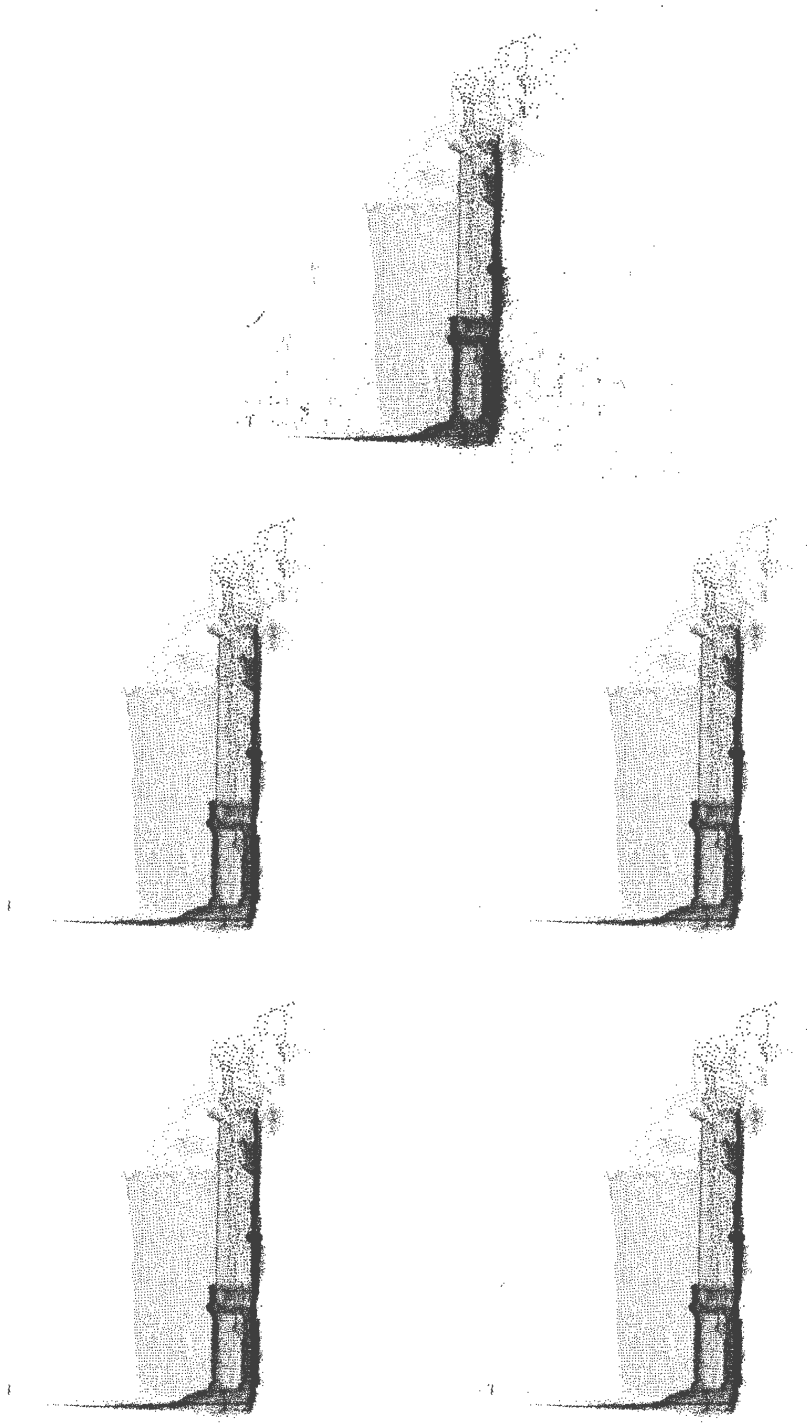


Figure 6.6: Point clouds of Entry-10. First row, left to right: initial point cloud, vertices in the final mesh of DC method with $th = 1K$, $th = 5K$. Second row, left to right: vertices in the final mesh of DC method with $th = 20K$ and vertices of global method.

final meshes.

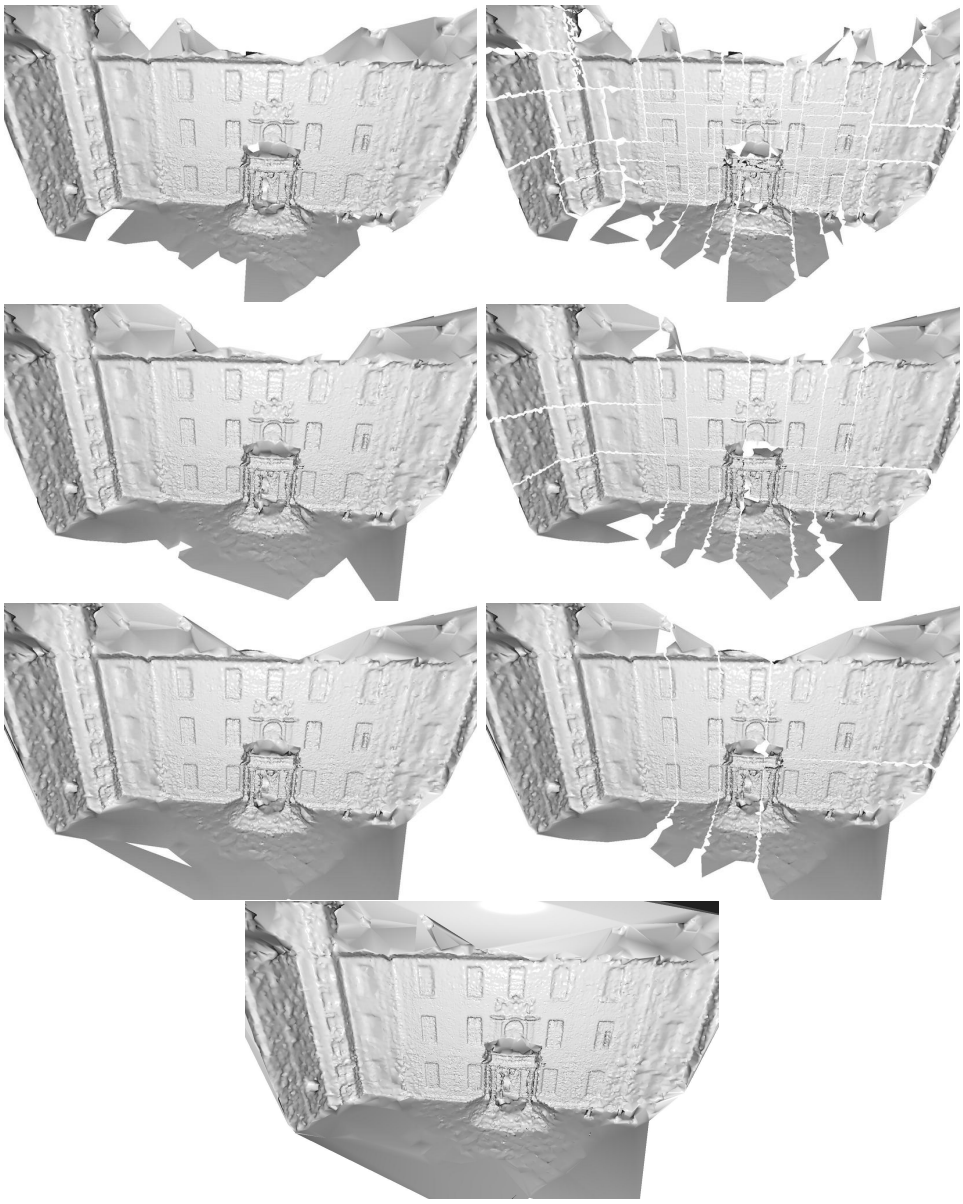


Figure 6.7: Result of the final surface with different threshold $th = 1K, 5K, 20K, \infty$ for each row. Each row, except the last one: left: union of the interior part of each sub-mesh before merging, right: the final mesh.

The tables Tab. 6.1 and Tab. 6.2 shows the running time, consumed memory, and other informations of each data set corresponding to different threshold (the final row corresponds to $th = \infty$, that we replace with input size). We notice that different thresholds do not have too different the running time. For Chamonix, the method runs fastest with $th = 500K$, while the global method ($th = \infty$) runs fastest in Lausanne point cloud. We can not deduce an optimal threshold for the speed, because it depends strongly on the size of input point as well as the

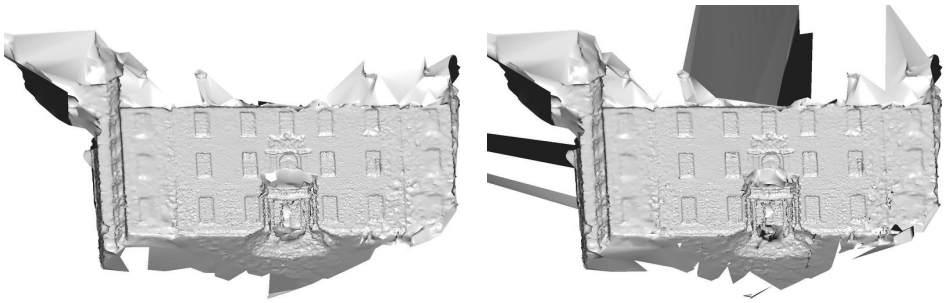


Figure 6.8: Final surface with and without multi-level point filter process ($th = 1K$). The surface without the filter can not remove outlier behind the scene.

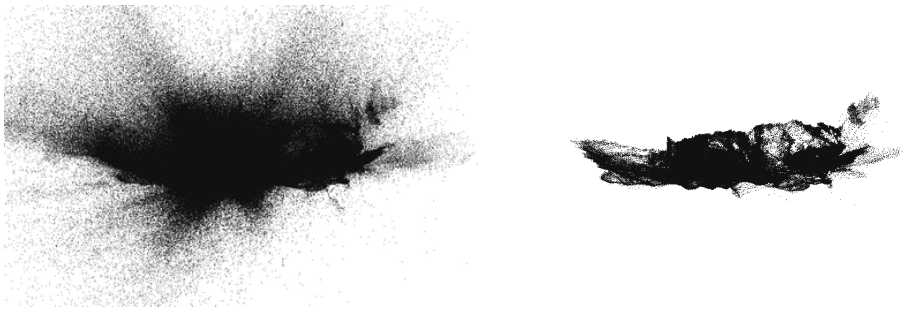


Figure 6.9: Filter outliers in Chamonix data. Left: initial input, right: vertices of the final surface in DC method with threshold of 200K.

number of line-of-sights. The consumed memory increases with th, due to the 3D Delaunay triangulation and the graph cuts nodes. The total memory used in global method make it unsuitable for a computer with less than 4GB (Chamonix: 3.7 GB, Lausanne: 4.5 GB, we need to count the memory for OS which is 500 MB or more). Please note that we store the whole point set with line-of-sights in RAM in the current implementation for convenient. However, it is not imperative. In the future, to handle with even large input point set that can not fit the RAM, we will modify the implementation to let the input in the hard disk. The only things must be in the RAM are the 3D Delaunay triangulation and the graph cuts optimization for each sub-set.

Incore size	# points /# facets	# subsets	RAM	time
10K	0.97M/2.00M	181	890 M	2915 s
50K	0.98M/2.01M	37	930 M	2394 s
200K	0.98M/2.01M	10	1.1 GB	2127 s
500K	0.97M/2.01M	4	1.4 GB	1779 s
2.650M	0.98M/2.01M	1	3.7 GB	2654 s

Table 6.1: Information of Chamonix point set.

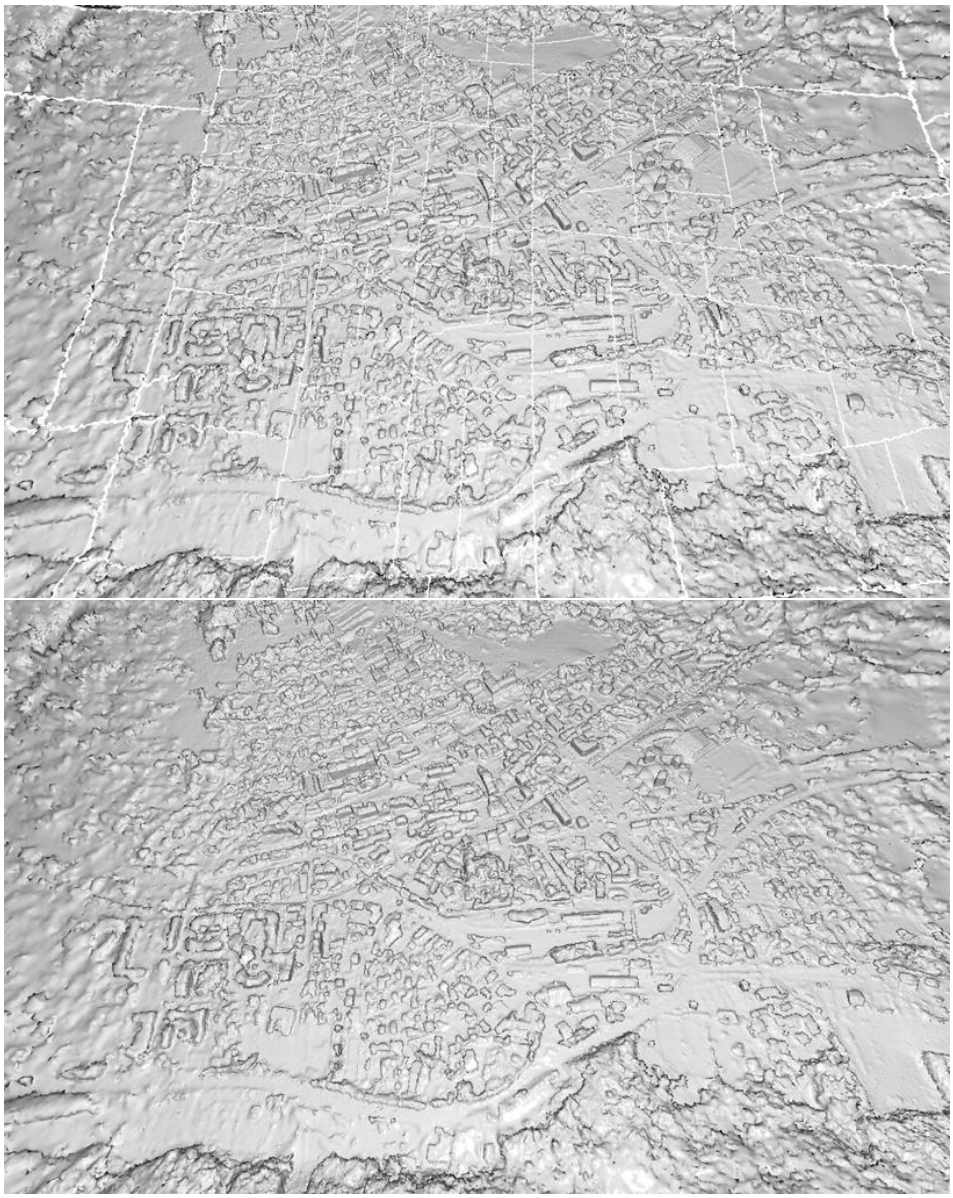


Figure 6.10: Urban zone in the final mesh of $th = 10K$. Above: partial surfaces inside each box of the bounding box partition. Below: final merged mesh.

Incore size	# points /# facets	# subsets	RAM	time
10K	1.45M/2.99M	261	470 M	1746 s
50K	0.98M/2.01M	53	515 M	1425 s
200K	1.45/2.99M	14	685 MB	1347 s
500K	1.45M/3.00M	5	1.0 GB	1332 s
3.054M	1.50M/3.01M	1	4.5 GB	1021 s

Table 6.2: Information of Lausanne point set.

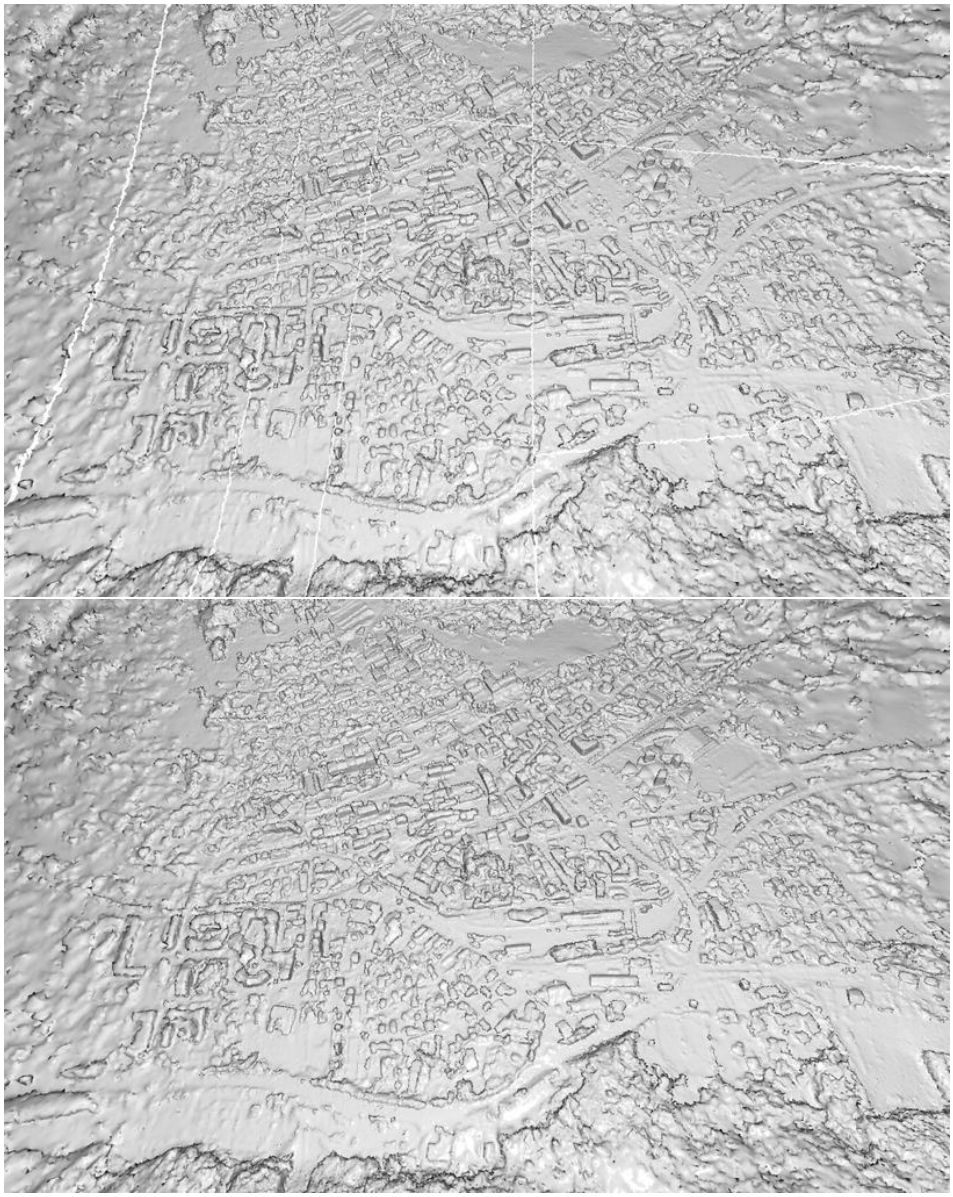


Figure 6.11: Urban zone in the final mesh of $th = 200K$. Above: partial surfaces inside each box of the bounding box partition. Below: final merged mesh.

Figure 6.12: Global visibility-consistent surface, corresponding to $th = \infty$.

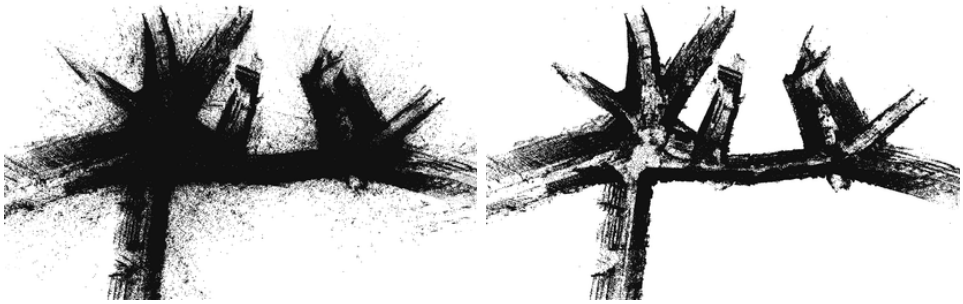


Figure 6.13: Filter outliers in Lausanne data. Left: initial input, right: vertices of the final surface in DC method with the threshold of 200K.

6.4 Limitation

Although we carefully design this DC approach, it still has 2 drawbacks. First, while it removes aberrant points of a noisy point set effectively, it can not remove all wrong triangles which are formed by correct points behind the ground truth . It is because these triangles do not cut other line-of-sights (no visibility constraints), the coarse-to-fine strategy only removes aberrant points (and facets formed by those points) but not wrong facets from good points. Second, although our merging method is effective and robust, it can let few small holes appear in the resulted mesh if the input meshes do not well consistent geometrically. These holes are more visible if the mesh is then smoothed or deformed (*e.g.* by a photometric variational method). To remedy these limitations, we will look at the energy formulations in both surface reconstruction and merging algorithms carefully.

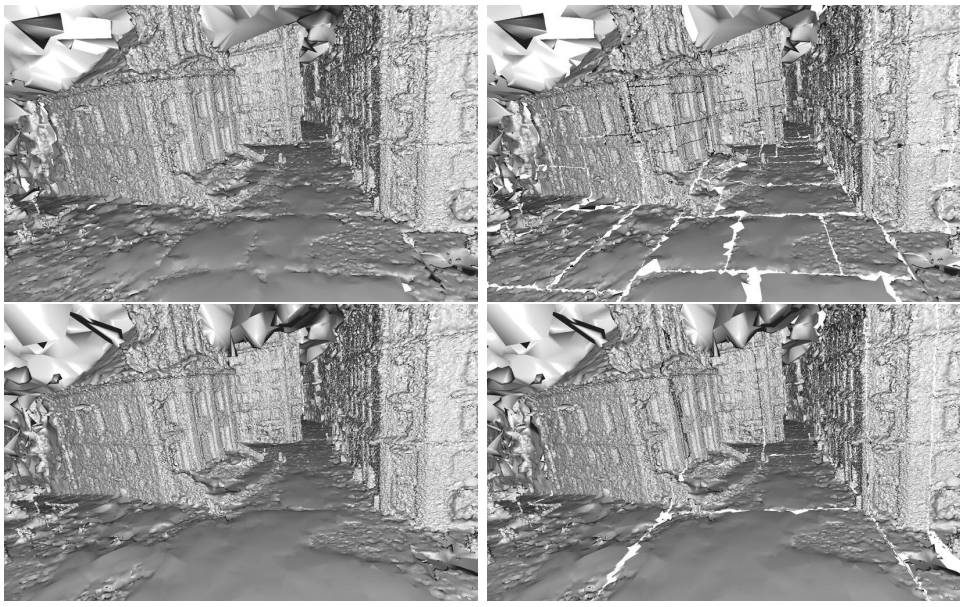


Figure 6.14: Lausanne data-set mesh with the threshold $th = 10K$ (first row) and $th = 200K$ (second row). Left images show merged surfaces, right images show partial sub-meshes inside boxes of the bounding box partition.

6.5 Conclusion

In this chapter, we have presented a Divide and Conquer approach based on the global visibility-consistent reconstruction method. It consists of 2 principal steps: the removal of outliers and the reconstruction of partial surfaces (followed by the merging process in chapter 5). It enables us to build a surface from a set of million tracks. This DC approach can profit of multi-core architecture or network computing in the surface reconstruction from each subset of the input although we have not implemented yet. In the future, we will try to apply other methods such as [Pan et al., 2009], [Lovi, 2010] into the surface reconstruction in our Divide and Conquer framework.

Chapter 7

Large scale multi-view stereo

Contents

In this chapter, we study the multi-view variational method of an initial mesh with a large number of high-resolution images, which is the extended version of the last step of the pipeline presented in chapter 4. We first analyze why large input data is an obstacle for this method. Second, we propose a Divide-and-Conquer approach to alleviate this issue. We show some experiments with some big data-sets to valid our approach. The main contribution of this chapter is the fair partition of the initial mesh into many fragments: each one associates with a set of sub-images, which can fit to GPU memory. We modify the variational implementation to take into account the visibility issue when refining each partial mesh. This approach is parallelizable.

7.1 Introduction

The multi-view pipeline presented in chapter 4 consists of 3 steps: plane-sweeping generation of point clouds, visibility-surface reconstruction, photometric variational method. The first step (point cloud generation) is truly scalable because in each instance, it only loads 2 images from the hard disk, to generate depth maps. The second step (surface reconstruction) is not quite scalable, and it has been specifically treated in chapter 6. This chapter will handle the last step, photometric variational method.

We recall the method's description in chapter 4. The input consists of images with cameras and an initial mesh. The mesh is deformed by photo-consistency gradient and regularization force, in order to minimize a given error projection:

$$\mathrm{E}_{\mathrm{error}}(S) = \sum_{i,j} \int_{\Omega_{ij}^S} h(I_i, I_{ij}^S)(x_i)\, \mathrm{d}x_i \tag{7.1}$$

where $h(I, J)(x)$ is a decreasing function of a photo-consistency measure between images I and J at pixel x (typically h is the opposite of normalized cross correlation) , $I_{ij}^S = I_j \circ \Pi_j \circ \Pi_i^{-1}$ is the re-projection of image I_j into image I_i induced by S and Ω_{ij}^S is the domain of definition of this re-projection (see Fig. 7.1), Π_i and Π_i^{-1} are the projection and back projection from an image i to the surface. This energy measures the sum of the dissimilarity between the portion of a reference image corresponding to the projected surface and a portion of another image re-projected via the surface into this reference image.

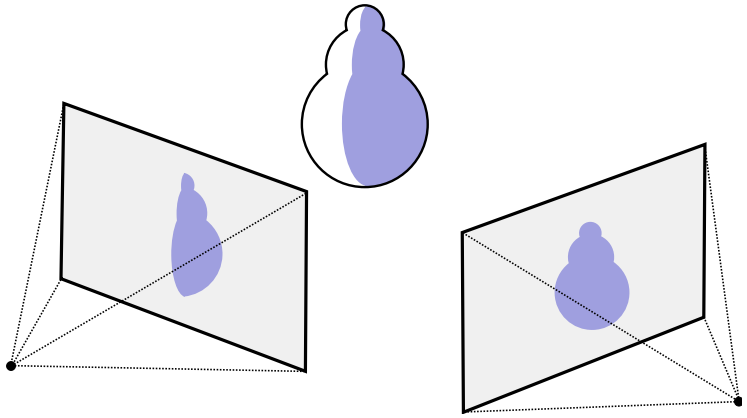


Figure 7.1: ***Reprojection induced by the surface.***

Because this energy takes over a set of images pair (i, j), then the gradient could be updated from each pair of image. In principle, we just to load many times a pair of 2 images from the hard disk into memory to compute their contribution to the gradient of vertices. However, different from the point cloud generation step, where each pair is loaded only one time, this variational step contains many iterations. Therefore, an image pair is loaded as many time as the number of iterations. Because hard disk access is slow comparing to in-core memory (except fast storage like Solid State Disk SSD), it is preferable to keep all images in in-core memory.

Our method is implemented for NVidia GPU acceleration, using OpenGL and shader language. Then for each pair of images, these 2 images and the mesh are transferred in GPU in-core memory (896 MB in a NVidia 260 GTX, 1.5 GB in a 480 GTX card) to run the computation. Sometimes, this GPU memory is quite small to contain all textures created from all input images, the program only stores a subset of images in the GPU RAM. The remaining data is stored in RAM. During the

computation, some textures are transferred from CPU memory to GPU memory if necessary. This displacement is done automatically by GPU driver that we can not control it.

In our implementation, for n images of max size $s = w \times h$, a total area of images a (then $a \leq ns$), a mesh of m facets, the total required memory size is a linear function of a, s, m: $f(a, s, m)$. We estimate $f(a, s, m) = 9.34a + 40s + 52m$ for color images and $5.34a + 35s + 52m$ for intensity images. For a data-set like Herzjesu-25 (a data-set of Strecha), which contains 25 images of 6 Mpixel, if the final mesh has the $2M$ facets, the memory needed is : $6(9.34 \times 25 + 40) + 52 \times 2 = 1745MB$ for color images. Hence, for a GPU's memory less than this amount, images textures will be moved many times from RAM to GPU memory. For even larger data-set, Chamonix, containing 1595 images of 5 Mpixel, this amount arises to 75 GB / 43 GB (for color / intensity images). That can not even fit to the memory of a typical workstation.

One insight is that the gradient of a point on the surface depends mostly on its surrounding area of the surface and local windows surrounding its projections in images. Hence, if the mesh is cut into small sub-meshes, and each sub-mesh is refined independently, the union of refined sub-meshes would be nearly the same as a refined global mesh. Moreover, refining a small sub-mesh does not require to load all input images, but only sub-images that containing the projection of the sub-mesh. That leads to the principal idea of this chapter: division the mesh into small sub-meshes such that the memory for each sub-mesh and its associated sub-images is smaller than a given amount of memory.

There are some previous papers working in the division of multi-view data-sets, but they targeted a slightly different issue: overlap viewing clustering, which extract many clusters of cameras from a large collection ([Zaharescu et al., 2008], [Furukawa et al., 2010]). [Zaharescu et al., 2008] divided the set of camera in each subset of cameras with k-mean, and for each subset, they collected a sub-mesh that can be viewed by this subset of camera. It is difficult to reduce image size, because the sub-meshes are created based on camera clusters. Therefore, in order to use sub-images in high-resolution data-sets, each image is replaced with multiple (typically 4) virtual images which capture many parts of the original image, before running the clustering method. Using a number of subsets as input parameters, they did not guaranty that each subset can fit a given limited memory. Our method does not divide the set of cameras, but cut the mesh into small fragment such that each fragment of the mesh will decide the associated sub-images. Hence, we have almost no trade-off between mesh quality and the memory size (or number of sub-data). With slightly different objective, [Furukawa et al., 2010] worked with a large collection of cameras with SfM point cloud. They divided the camera set to

meet many criteria: compactness, size constraint and coverage. They also proposed a Multi-view filtering (quality and visibility filter) for point cloud generated by PMVS algorithm ([Furukawa and Ponce, 2008]). [Jancosek et al., 2009] produced separated depth maps (and meshes) for each camera (in removal redundant or similar views in a data-set). This approach shares the same scalability as the first step of our pipeline (depth map generation). Nevertheless, the visibility issue is treated incrementally. The obtained result is a set of separated meshes which are not refined to get a better quality. Different from them, our algorithm yields a topologically connected and highly accurate mesh for large data-sets.

In principle, our whole pipeline reconstruction does not require a camera clustering formulation to produce a large scale reconstruction. We recall that, its first step, the depth map generation, is naturally parallelizable and scalable. The second and the last step require a division of the point cloud (chapter 6) or the initial mesh (this chapter) into small parts before collecting associated cameras and sub-images. Nevertheless, we are aware that for extremely large scale data-sets (more than hundred thousands images), it is preferable to cluster cameras into many isolated clusters to reduce the running time.

7.2 Partition of mesh with associated images

This division of an input mesh into small sub-meshes is somehow similar to the point cloud partition in chapter 6. The main difference is that the current problem takes into account the size of of sub-images who contain the projections of each sub-mesh. Similar to chapter 6, we divide the mesh with hyperplane perpendicular to a coordinate axis. A sub-mesh related to a bounding box is the set of all facets having at least one point inside this box. Therefore, when a mesh is divided by a partition of the bounding box method, we obtain a set of slightly overlapped sub-meshes.

The problem is stated as: "Given a mesh M inside a bounding box B with n calibrated images $I_1, \ldots, I_n$ of size $w \times h$. Given a consumed memory formula f depending on size of n images: $w_i \times h_i$, number of facets m of the mesh, a threshold th. Partition the bounding box B into small bounding boxes, such that each sub-mesh contained in each bounding box, associated with its sub-images, has its consumed memory (given by the formula f) not greater than th. The number of boxes/sub-meshes should be small".

The sub-image containing the projection of a sub-mesh in an image I is the smallest rectangle of I that contains this projection. The division of mesh by the bounding box partition does not lead to a simple partition in an image domain. The number of vertices, facets of a sub-mesh have no direct relation with the size

of sub-images.

To handle this problem, we use the following sub-routine: "partition the bounding box B into 2 bounding boxes, such that the amount of memory for each sub-mesh and its sub-images, given by the formula f are nearly equal". After solving this sub-routine, we successively divide each bounding box by two until the consumed memory for each sub-mesh not greater than th. At the end, we then obtain a binary tree of bounding boxes, in which the leaves correspond to sub-meshes and sub-images whose consumed memory does not exceed th.

For this, we suppose the bounding box of the input mesh is $B = [0, 1]^3$, which is divided by a hyperplane perpendicular to x axis at position $x = u$. $g(u)$ is the total necessary memory for the sub-mesh between $x = 0$ and $x = u$, $h(u)$ is the total memory for the sub-mesh between $x = u$ and $x = 1$. We have $g(0) = h(1) = 0$ and $g(1) = h(0)$. For $0 < u_1 < u_2 < 1$, the sub-mesh and sub-images between $x = 0$ and $x = u_1$ are contained in the sub-mesh and sub-images between $x = 0$ and $x = u_2$, then $f(u_1) \leq f(u_2)$. Hence, g is increasing, similarly, h is decreasing, then the difference $d(u) = g(u) - h(u)$ is a decreasing function from $-h(1) < 0$ to $g(1) > 0$. We want to find u such that $d(u) = 0$ or nearest the zero.

One common solution is using *bisection method*: first, test the value $u_1 = 0.5$: $d(0.5)$. If $d(0.5) > 0$ then the optimal u would be in $[0, 0.5]$, and the search in repeated in $[0, 0.5]$. If $d(0.5) < 0$ then the optimal u would be in $[0.5, 1]$ and the search is repeated in $[0.5, 1]$. For each segment, the middle value is tested. We then have the sequence: $u_1, u_2, \ldots$. The search is stopped when $d(u_k)$ or $|d(u_k) - d(u_{k-1})|$ near to zero. This algorithm is practical if the computation of the functions g and h is cheap.

In our case, the estimation of g and h evolves the computation of depth of the sub-mesh with images (to handle the visibility) and the projection of sub-mesh in each image. The multiple estimations of $d(u)$ in bisection method require many repeated projections, which are expensive. Therefore, we use a more direct approach. First we estimate the depth map of the whole input for into each image only one time to obtain n depth maps: $D_1, \ldots, D_n$. Second, we consider a fixed sequence of value u: $0 = u_0 < u_1 < \cdots < u_k = 1$ and compute all $d(u_i)$ $(1 \leq i \leq k)$ to choose the smallest $|d(u_i)|$.

By using precomputed depth maps, we traverse all pixels in depth maps only one time to compute all $d(u_i)$. For each image I with depth map D, we traverse all pixels, each pixel p corresponds to a 3D point P of the mesh. Considering x axis of P : $P.x$, for all i, j such that $u_i \leq P.x \leq u_j$, we conclude the point P will be on the sub-mesh of I between $x = 0$ and $x = u_j$, and between $x = u_i$ and $x = 1$, that will update the value of $f(u_j), g(u_i)$. After traversing all pixels of all images, we can compute all $d(u_i) = f(u_i) - g(u_i)$. We choose the i such that $|d(u_i)|$ minimum.

For implementation, we consider the sequence u of $k = 16$ elements such that each slice $x = [u_i, u_{i+1}]$ contains $1/k$ number of vertices of the initial mesh.

The variational method runs in multi-level, which takes a refined mesh from images of lower resolution, subdivides and deforms it with higher resolution images. We do not need to compute this binary tree of boxes for each resolution, we only compute one time in the beginning with input mesh and full resolution images. In each resolution, we travel from the root, and choose boxes which have the required memory not greater than th, otherwise, we consider their two child boxes. We then obtain for each resolution, a set of small boxes, corresponding to a set of sub-meshes to define.

7.3 Sequential and independent deformations

Once we divide the mesh into many sub-meshes with its associated sub-images, the next step is running the variational method for each. We could deform each sub-mesh in a sequential (serial) or an independent algorithm (parallelizable).

In a sequential implementation, each sub-mesh is deformed in order without separating the whole mesh. It is like the refinement of the input mesh in different regions over time. Its main advantage is to keep the topological connectivity of the mesh. If the mesh is refined within one computer, a sequential implementation would be the best choice because there is no problem with the mesh topology. Because our algorithm use intensively GPU computing and memory, it can not use many CPU-cores for parallelization within a computer.

If we have several available machines, we had better to cut the sub-meshes (with its sub-images) from the initial mesh, and refine them independently in these machines to reduce the computation time. During the deformation, common vertices of different sub-meshes change their position and do not stick in the end. The situation is more complicated when each sub-mesh is subdivided independently because we can not simply match additional vertices. We need to merge them together from a partition of bounding box (chapter 5).

In fact, there is a simple alternative to merge sub-meshes in this case. In order to avoid matching subdivided vertices of different sub-meshes, we can subdivide the whole mesh before partitioning it. Next, we mark all vertices strictly inside each box, associated with each sub-mesh. Therefore, while a vertex can belong to many meshes, it has only one mesh 'native' to decide its final position. After deforming each sub-mesh related to each box, we update new position of all vertices. With available connectivity information, we obtain the whole refined mesh. With a careful implementation, this approach could work in parallelization for different machines. It is better than applying the mesh merging because it is like a sequential

deformation in a parallel manner. However, this idea comes to us very recently so that we have not implemented it yet. Please note that in case there is no connectivity information, only a merging method (chapter 5) can be effectively used.

Another concern about the deformation of a sub-mesh is its visibility in relation with other sub-meshes. Given a sub-mesh $\mathcal{M}_0$ from the input mesh $\mathcal{M}$, a sub-image I_0 of an image I containing the reprojection of $\mathcal{M}_0$. A point P of $\mathcal{M}_0$ may have its projection inside the domain of I, however, if P is hidden by the rest of mesh $\mathcal{M}$, this projection pixel does not correspond to P. Hence, the computation of projection has to involve a depth test to detect that occlusion. Therefore, after computing the sub-image I_0, we still need to mark pixels of I_0 that does not correspond to $\mathcal{M}_0$. Otherwise, the back reprojection of I_0 into $\mathcal{M}_0$ could be wrong.

7.4 Experiments

We test with 3 data-sets: Cluny-26-Big (26 grey images of 56 MPixel), Cluny-161-Small (161 grey images of 5.3 MPixel) and Chamonix (1596 grey images of 5.3 MPixel) associated with their initial meshes. In fact, the two last data-sets have original images of 21 MPixel but our calibration of cameras is not accurate enough to build a better 3D model at this resolution. Thus, we reduce its resolution by two in each dimension. Using intensity or color images does not produce a noticeable change in our variational method results, we convert them into grey images to economize the memory. The initial meshes were reconstructed with our visibility-consistent reconstruction methods (Cluny meshes: original method in chapter 4, Chamonix mesh: DC method with the threshold of 200K in chapter 6).

All experiments are done in a workstation with 2 CPU 4 cores of 2.6 GHz, 24 GB RAM and a Nvidia 480 GTX GPU. Because our method relies essentially on GPU, the performance of CPU and the large capacity of RAM are not important. The Nvidia 480 GTX GPU has 448 cores of 1.5 GHz and 1.5 GB Memory. Our variational method has a parameter th which indicates the maximum allowed consumed GPU memory. This parameter decides the division of the input mesh into sub-meshes (section 7.2). We use $th = 1GB$ for all data-sets, except another experience conducted with $th = 512MB$ for Cluny-26-Big data-set. Because we use only one machine, our method is implemented in the sequential manner, except a test is applied for Cluny-26-Big. The following table lists input information of 3 data-sets:

Data-set	# images/pairs	Resolution	# total pixels	#initial facets
Cluny-26-Big	26/102	5992×9326	1.45 GPixels	889K facets
Cluny-161-Small	161/494	2808×1872	836 MPixels	480K facets
Chamonix	1595/5316	1875×2812	8.41 GPixels	747K facets

Firstly, we consider Cluny-26-Big data-set, 26 images of 56 MPixel taken from a balloon. The variational method refines the mesh from a set of mipmap images, from level 4 to level 0 (full resolution). The input memory size increases from the highest mipmap level to the lowest one. Therefore, each level has a different number of sub-meshes. In order to verify the impact of GPU memory threshold, we consider 2 cases: $th = 1GB$ and $th = 512MB$. We do not see any quality difference in the final mesh in 2 cases (Fig. 7.3). To see the variational process in different levels, Fig. 7.4 shows the chapel of Cluny Abbey from initial mesh to multi-level refined mesh.

The below table describe the running time and number of partition sub-mesh in each mipmap level. Some table cells contain 2 values, where the first value corresponds to $th = 1GB$, the second corresponds to $th = 512MB$.

Cluny-26-Big	#sub-meshes (1.0/0.5 GB)	# output facets	times
binary box tree			10 / 20 s
mipmap-4	1	916 K	1.3m
mipmap-3	1	974 K	1.6m
mipmap-2	1 / 3	1.42 M	2.3m / 2.5m
mipmap-1	8 / 24	4.74 M	8.9m / 9.7m
mipmap-0	43 / 85	18.3 M	37m / 43m

In this above table, the number of sub-meshes of $th = 512MB$ is quite greater than that of $th = 1GB$ in full resolution (85 vs 43) without a big impact on the running time (43m vs 37m). In fact, it is not a surprise because the variational method complexity depends on mesh size and its projections on input images (with some margins to compute correlation window and anticipate mesh projection change). No matter how we divide the mesh, this quantity should be almost the same. From this, we can argue that the memory-equal division of a sub-mesh into 2 sub-meshes in section 7.2 does not change the running time significantly. Nevertheless, this division allows similar quantity of data for each sub-mesh, which will be useful for parallel computing. We also notice that the bounding box tree computation time is negligible comparing to the total running time (20s vs many minutes).

Beside this sequential deformation, we test the independent deformation of sub-meshes in mipmap-1 resolution (with $th = 1GB$): 8 sub-meshes are deformed independently and are merged at the end. The surface quality remains the same, and visually, it is difficult to distinguish each sub-mesh unless we zoom in an overlapped zone (Fig. 7.5). The merging process (section 5.3) takes 15 seconds (I/O time from

Figure 7.2: Cluny-26-Big datasets with 26 images of 56 MPixel. Left: initial mesh. Right: final mesh in the full resolution with $th = 1GB$.

hard disk included) and successfully merge them together. To compare the quality of serial and independent deformation, we apply 30 Laplacian smoothness operations in Fig. 7.6: the quality of meshes in two cases are almost the same, while the simple union of sub-meshes crack as expected. While using this merge method is not the only way to combine refined sub-meshes (as previously discussed in section 7.3), this experiment shows again the robustness of the merging method.

The data-set Cluny-161-Small has more number of images, in much lower res-

Figure 7.3: Cluny-26-Big refined mesh with memory threshold $th = 1GB$ and $th = 512MB$. There is almost no difference in 2 meshes.

olution (5.3 MPixel vs 56 MPixel). As we said, due to less accurate calibration, we do not use full resolution 21 MPixel of the original image. We obtain a more complete mesh of Cluny Abbey with less accuracy (Fig. 7.7). The following table displays computation information for this data-set:

Cluny-161-Small	# sub-meshes	# output facets	times
binary box tree			23 s
mipmap 2	1	506 K	3m
mipmap 1	2	779 K	4.5m
mipmap 0	14	2.7 M	16m

The biggest data-set is Chamonix, consist of 1596 images of 5.3 Pixels, taken from a helicopter. The scene is so vast that we show only a small part of it in initial and final refined mesh (Fig. 7.8). We show computation information in the following table:

Chamonix	# sub-meshes	# output facets	times
binary box tree			7 m
mipmap 2	4	2.6 M	35 m
mipmap 1	22	8.9 M	1.1 h
mipmap 0	110	36.3 M	4.2 h

The next table displays information of previous steps of the multi-view pipeline (point cloud extraction and visibility-consistent surface reconstruction). Because we did not use exactly the same computer as this variational step, the running time is just for reference. Please note that after the surface reconstruction step, we automatically remove exceedingly large facets, small isolated connected components, and manually cut additional facets outside an interested area, before running the photometric variational step. The number of images and pairs for Chamonix data set in the point cloud extraction step (2118/19400) is larger than the variational step (1596/5316) because we want to refine only the urban area of the scene.

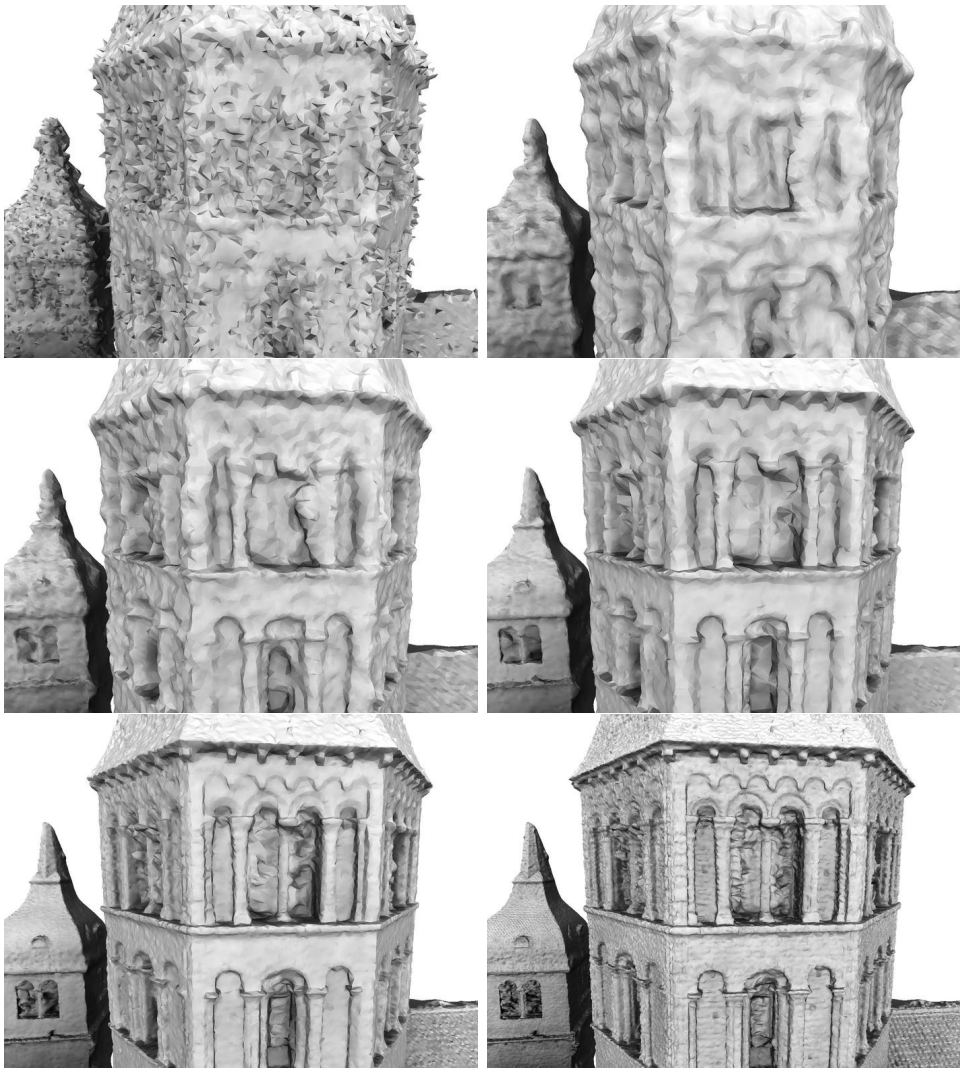


Figure 7.4: Cluny-26-Big (26 images of 56 MPixels) in different resolution. From up to down, left to right, detail on input mesh, refined mesh with the mipmap 4,3,2,1,0 (the final mesh). They have 889 K, 916 K, 1.42 M, 4.71 M, 18.3 M facets respectively.

Data-set	#images/pairs	Mipmap	# 3D points/time	# surface facets/time
Cluny-26-Big	26/102	2	1.03 M / 17.7 m	1.26 M / 8.9 m
Cluny-161-Small	161/494	2	350 K / 38.7 m	530 K / 2 m
Chamonix	2118/19400	2	2.65 M / ∼10 h	2.01 M / 35.4 m

7.5 Some comparison with PMVS

Patch-based Multi-view Stereo Software (PMVS) [1] is a 3D reconstruction software of Yasutaka Furukawa and Jean Ponce, which is the implementation of their method

[1] `http://grail.cs.washington.edu/software/pmvs/`

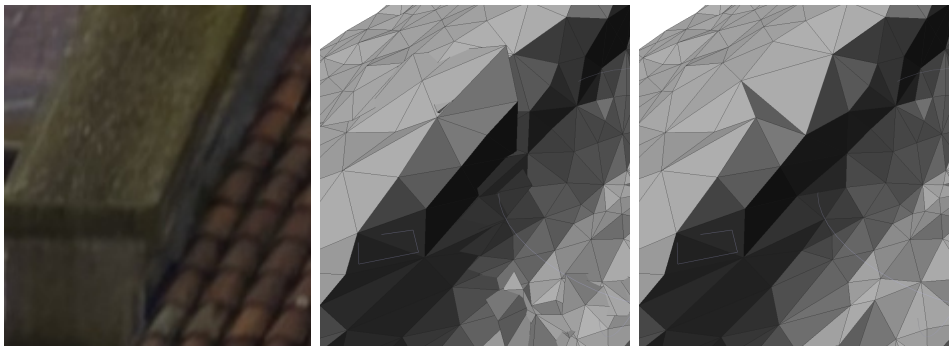


Figure 7.5: Cluny-26-Big: A simple union of meshes and the merged mesh in the independent deformation. Because the scene is too big to distinguish different sub-meshes, we only zoom in a detail, found on a roof of the main building.

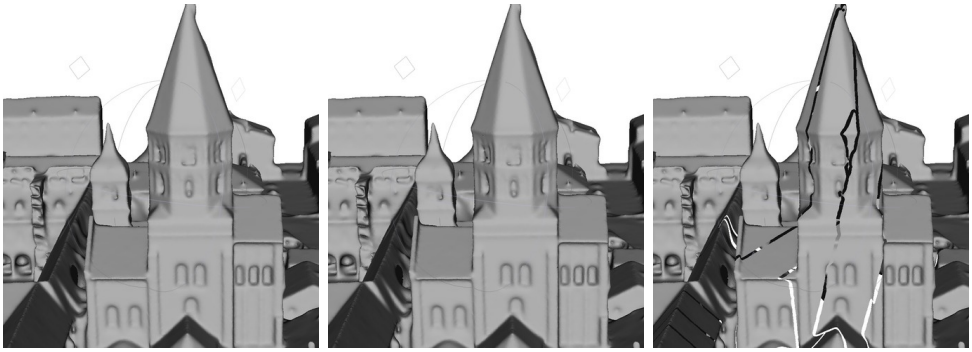


Figure 7.6: Cluny-26-Big: (Level 1) refined meshes are smoothed with 30 Laplacian operations. Left to right: mesh with serial refinement, mesh is merged from 8 partial refined sub-meshes, union of these 8 refined sub-meshes. There is visually no crack in merged mesh, which prove the success of the merging algorithm.

(conference version [Furukawa and Ponce, 2007] and journal version [Furukawa and Ponce, 2008]). We use the Windows version ported by Pierre Moulon [2] . The software takes as input the images and calibrated cameras, and produces a set of patches or a point set with normal vectors, eventually with line-of-sights from cameras. PMVS achieved the best result (both completeness and accuracy) for 4 / 6 data-sets of Strecha in 2007. Lack of a variational refinement step described in their paper, PMVS does not provide a full implementation of the method, hence the comparison of our multi-view pipeline and PMVS is not complete. Because it produced highly accurate point sets, we are curious to see whether it is possible to use its point clouds in our pipelines. We also compare the meshes of our method to meshes reconstructed from PMVS patches (with Poisson reconstruction and the visibility-reconstruction surface method). We will use 2 data-sets: our Cluny-26-Big, and the Hall data-set (61 images of 6MPixel) in PMVS package. For Hall

[2] `http://francemapping.free.fr/Portfolio/Prog3D/PMVS2.html`

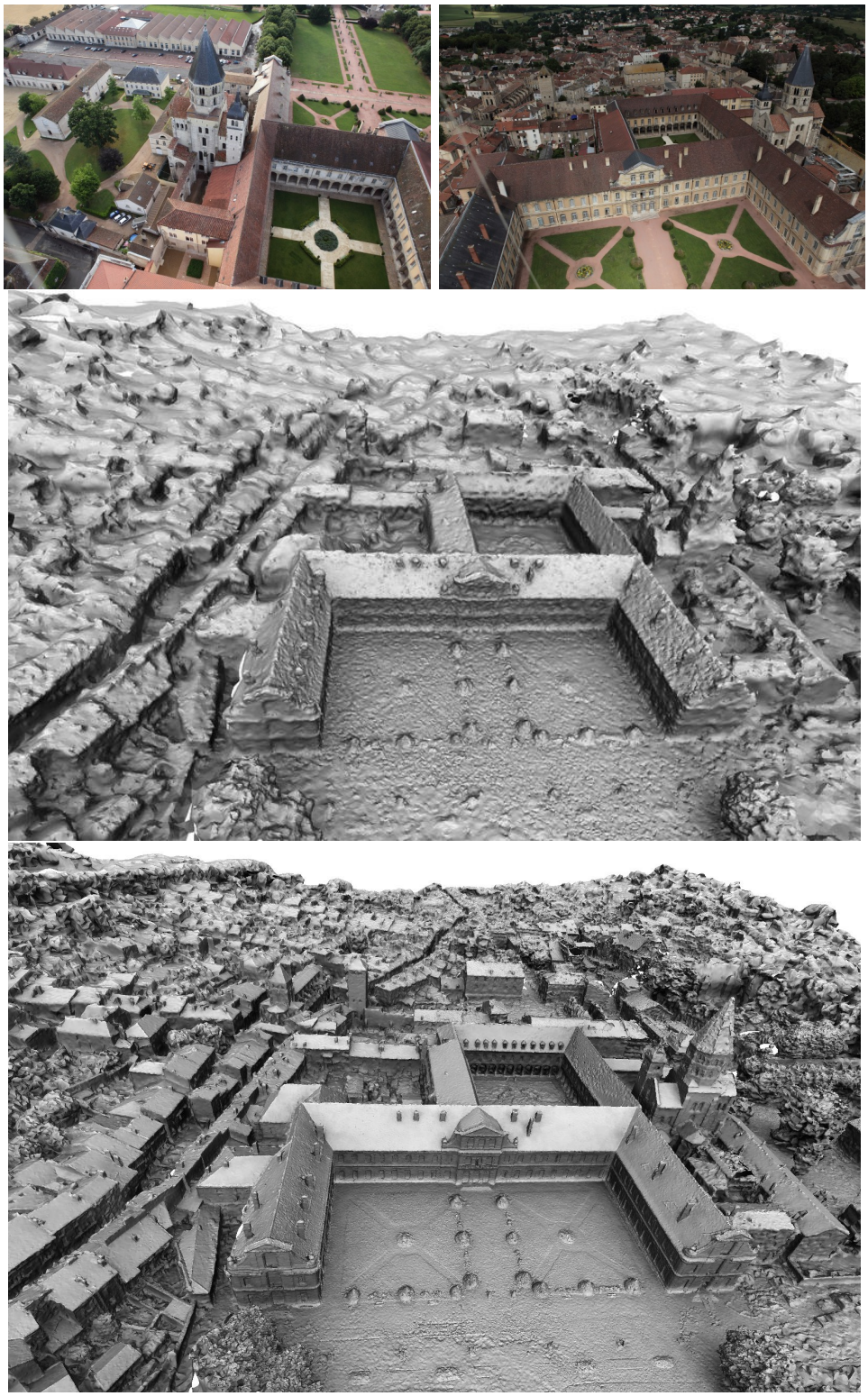


Figure 7.7: Cluny-161-Small (161 images of 5.3 MPixels). Initial mesh (480 K facets) and final refined mesh (2.6 M facets).

Figure 7.8: Chamonix (1595 images of 5.3 MPixels): initial mesh (747 K facets) and final refined mesh (36.3 M facets).

data-set, we use 120 pairs of neighboring images.

Although PMVS does not use a GPU acceleration, it can exploit all 8 CPU cores of our machine for the computation. That makes it have the same order of performance as our plane-sweeping point cloud generation: $10m$ vs $17.7m$ for Cluny-26-Big and $14m$ vs $7m$ for Hall data-sets. Because PMVS aims to create an accurate point set, it has the trade-off with the scene completeness: its Cluny-26-Big point cloud is less complete than ours (Fig. 7.9). It captures main features of the scene but it is still not free from outliers. Due to false matching of pixels of the sky and, and accentuated by the belief propagation (false matching seeds propagate), it obtains denser false floating points than our point cloud.

We run the visibility-consistent surface reconstruction for two point clouds, then remove facets outside a given bounding box and too big facets, to obtain the initial

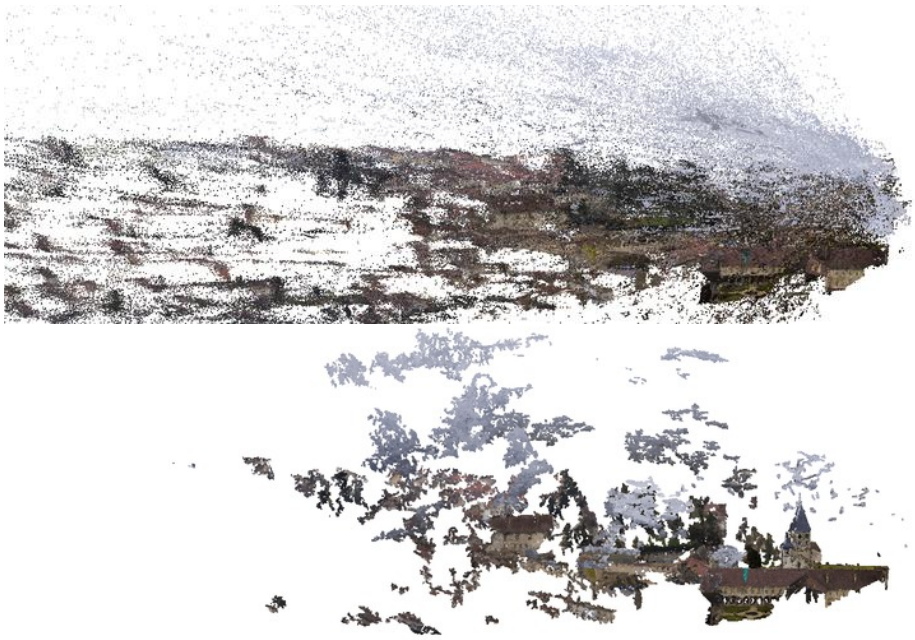


Figure 7.9: Point cloud generated by our GPU sweeping-plane method (up) and PVMS2 (bottom). PMVS point cloud has less noise but also less complete.

mesh for variational method in Fig. 7.10). This surface reconstruction does not remove false floating points in PMVS point set, because there is little visibility constraint to remove them. Our point set does some false sky points, but they are less dense, and there are much more light-of-sights (include light-of-sights from very far points) going through those false points. While the refined mesh from PMVS point cloud has equivalent accuracy as our original refined mesh, the floating outliers can not be removed.

The Hall data-set consists of 61 images around a hall, included in the PMVS package. The images are taken sequentially around the building (while images in Cluny-26-Big are much less controlled). We executed PMVS with the option file for this data-set, but we did not produce a point cloud as complete as the expected one (also available in the package). Therefore, we consider this available point cloud, which is nearly as complete as our plane-sweeping point set. Because a point / patch in PMVS output also can remember the images from which it is generated, we can extract a visibility-surface reconstruction from it (Fig. 7.12), and refine it with the variational method (Fig. 7.13). While initial mesh generated from point cloud of PMVS is much cleaner than from out point cloud, the refined meshes have the same excellent quality. We also observe that the Poisson surface mesh directly generated from the PMVS point set is significantly less accurate.

This short comparison shows the advantage and disadvantage of our GPU sweeping-plane point cloud extraction and the current PMVS. The two approaches

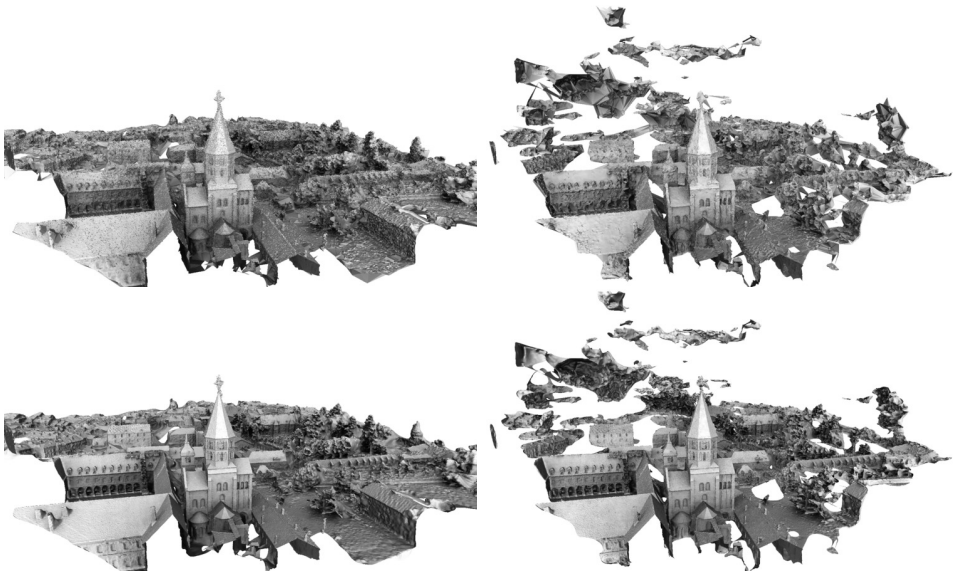


Figure 7.10: First row: initial mesh from our point cloud and from PMVS cloud. Second row: associated refine meshes with mipmap level 2.

Figure 7.11: Hall data-set. Top: 3 among 61 input images. Bottom: from left to right: our point cloud, the PMVS point cloud found in the package, the PMVS point cloud which we obtain.

take advantage of parallel computing (GPU vs multi-core CPU) to have high performance. Taking care of accuracy, PMVS produces less complete point clouds then our method. Comparing the Poisson mesh from an accuracy point cloud of PMVS and the refined mesh from our pipeline, we confirm that a photometric variational is vital for a high quality reconstruction.

Figure 7.12: Visibility consistent mesh with our GPU sweeping-plane point cloud (left) and PVMS point cloud (right). The PMVS point cloud leads to more accurate initial mesh.

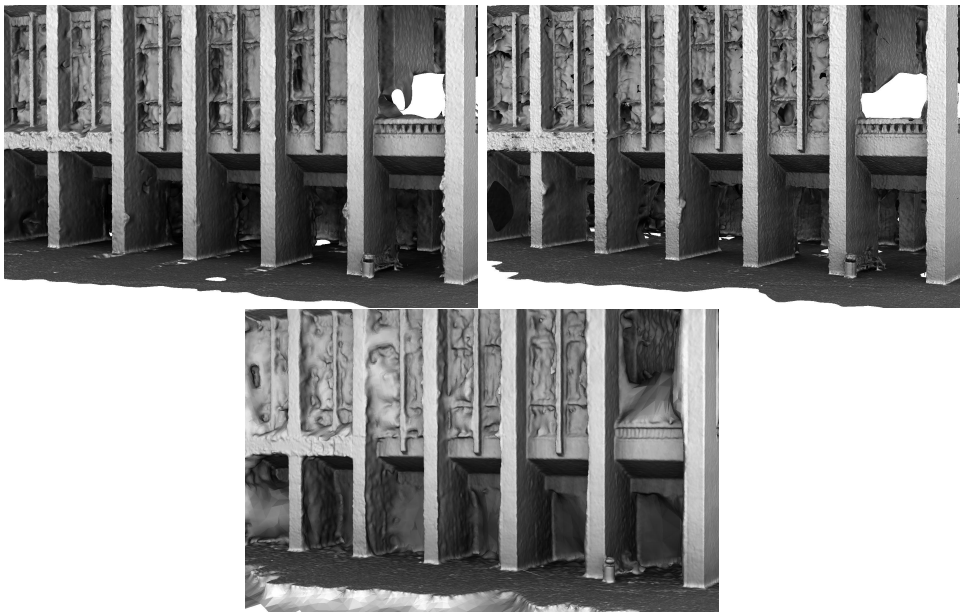


Figure 7.13: Refine meshes from initial meshes in Fig. 7.12 (the top 2 images), and mesh generated by Poisson surface reconstruction. Without a refinement process, Poisson surface is not quite accurate.

7.6 Conclusion

This chapter presents a variational method enhanced with memory-equal partition of mesh, in order to handle very large data-sets that original method can not handle. The amount of GPU memory decides the sub-meshes number in each level of image resolution, however, it has no impact on mesh quality and slightly overhead on the running time. This flexibility allows us to apply the variational method to build large-scale meshes of high quality. However, this quality depends largely on the calibration accuracy, which is also a challenging issue in computer vision. In the future, we will deploy this algorithm to distributed computing over a network. The variational method is applied for each sub-mesh independently in each machine and

the final refined sub-meshes are merged with a method presented in section 5.3, in order to produce the same quality mesh with less running time.

This chapter with previous chapters 5, 6 complete our pipeline (in chapter 4) for large-scale multi-views stereo. We recall the process to reconstruct a 3D surface from a set of calibrated images : extraction of point cloud, reconstruction the visibility-consistent surface (with large scale version in chapter 6), refinement of the initial mesh (with large scale version in this chapter). In fact, after the point cloud generation step, there is different possibility to build large-scale models: reconstruction a set of overlapped surfaces (as in chapter 6 but we do not merge meshes), refinement of these partial meshes followed by a merging process at the end.

Chapter 8

Conclusion

The central goal of this thesis is the reconstruction of high quality multi-view stereo from many of high resolution images. We have provided a high performance and quality multi-view stereo pipeline. We have developed scalable approaches in order to run this pipeline for large data-sets (up to thousands of 5 MPixel images), and have produced large geometrically and topologically accurate meshes. The thesis contribution can be resumed as follows:

- We have combined and improved previous works in order to obtain a multi-view stereo pipeline, that produces highly complete and accurate meshes. First, a point set with visibility information is extracted from photo-consistency matching over pairs of images. Second, a global visibility-consistent surface is extracted over this point set, with Delaunay triangulation and graph cuts optimization. Finally, an enhanced photometric variational method refine this surface to increase its accuracy.

- We have created a fast, robust and automatic merging method that merges multiple separated meshes into a single surface. It can take advantage of a bounding box partition to accelerate merging process if this information is available. The method uses Constrained Delaunay tetrahedralization and graph cuts optimization.

- We have developed a non-trivial Divide and Conquer approach for visibility-consistent surface reconstruction of a large, noisy point set with visibility information. This approach consists of 2 separated steps: remove outliers and surface reconstruction, which share many common components. To remove outliers behind the surface effectively, we work with multi-level representations of the point set. We use our merging method to combine partial meshes at the end. The computation of a surface or inliers in each part can be run in parallel.

- The photometric refinement of a mesh is limited by GPU memory. Then we have thoughtfully partitioned the mesh into many sub-meshes so that each sub-mesh associated with portions of images containing its projection, can be fit in GPU memory. Each sub-mesh is refined in an independent (parallel)

or a serial manner. In both cases, we obtain the same accurate result. This partition allows us to produce high quality and complete mesh from thousands of high resolution images.

A highlight feature of our multi-view pipeline is its high modularity. The plane sweeping point set extraction, the visibility consistent surface reconstruction, the photometric refinement and the mesh-merging method are independent and can be replaced by other similar methods. This independence might enable them to apply in other contexts.

We believe many points in the thesis could be further investigated and improved, such as parameter tuning, direct comparisons with other methods. More importantly, some additional majors issues should be addressed in perspective to boost the multi-view stereo:

- The main obstacle of our multi-view methods is the accuracy of calibration. Currently, we use the calibration software Bundler ([Snavely et al., 2006]), which can calibrate remarkably well for some data-sets, but not for all. The variational method, while produces highly detailed mesh features, is sensitive for calibration error. It is not a surprise because accurate reconstruction requires the same degree of calibration accuracy. The lack of accuracy in current state-of-the-art calibration is the main obstacle for us to limit the image resolution in the 3D reconstruction.

- Moreover, the variational method should identify areas of mesh that do not have a good calibration measure in order to apply an appropriate treatment. A more ambitious solution is to optimize mesh and camera calibration in some manner, similar to [Furukawa and Ponce, 2008], [Tylecek and Sara, 2009].

- Our methods, as many other classic multi-view stereo methods, suppose the 3D objects are more or less Lambertian. While it is almost true for ancient buildings, statues, etc, it is not true for many materials like glass, specular plastic. If it is difficult to reconstruct them successfully, at least, detecting and ignore them could be useful. The same argument is applied for moving objects (people, cars) across images.

- When there are many data-sets for the same 3D scene, but their cameras (and resulted mesh) are not in the same coordinates, it is advisable to merge them together (matching images and mesh) to obtain a bigger data-set. This could be helpful for the calibration or the refinement of some details of a mesh using supplement images, which are not in the current data-set associated with the mesh.

The ultimate goal of multi-view stereo and computer vision is to design algorithms and devices, which are intuitive and have the same capacity as human vision. From the achievements in the domains, we can think about saving the 3D models of humans, trees, buildings, animals with cameras or mobile phones instantly. That will bring immense applications in game, cinema, architecture, computer industry, not to mention the military. It is not a far future because multi-view stereo today are efficient enough for average users produce and enjoy their 3D models with free applications such as Microsoft's Photosynth[1] and Autodesk's 123DCatch [2]. The success of computer vision will help to create more intelligent robots which understand and recognize the environment in order to reduce human labor. For now, Google has experimentally tested a prototype of an auto driver [3]. Everything still just begins.

[1] http://photosynth.net
[2] http://www.123dapp.com/
[3] http://googleblog.blogspot.com/2010/10/what-were-driving-at.html

Appendix A

Background

A.1 Delaunay triangulation

A triangulation of a point set P in $\mathbb{R}^d$ is a partition of its convex hull into simplices of dimension d. In 3-dimension, it is also called *tetrahedralization*. A given set of points admits, in general, many triangulations.

A Delaunay Triangulation of a point set P is a triangulation with *empty disk* property: no point in P is inside the circumcircle of any simplex of this triangulation. In the general position, where there are no $d+2$ points on the same sphere, the Delaunay triangulation is unique.

A.2 Constrained Delaunay triangulation

Instead of an input point set P, we have a *Piecewise linear system* (a set of points, edges, triangles such that they do not cut themselves in the interior) as input, the similar concept of Delaunay Triangulation would be the *Constrained Delaunay Triangulation / Tetrahedralization* (**CDT**). The CDT shares many useful properties with Delaunay Triangulation while it contains the input constraint (with eventually additional points, called Steiner Points).

In two dimensions, CDT was first defined by Lee and Lin [Lee and Lin, 1986], Chew [Chew, 1989], and such triangulation always exists (Fig. A.1). However, in three dimensions, the CDT may not exist because not every 3-dimension polyhedron allows a tetrahedralization without additional vertices (eg. Shönhardt's polyhedron,

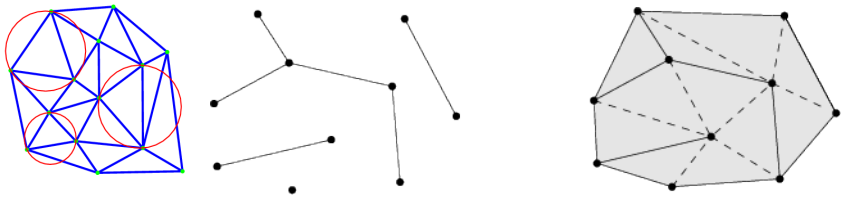


Figure A.1: Delaunay Triangulation, and constrained Delaunay triangulation (CGAL manual).

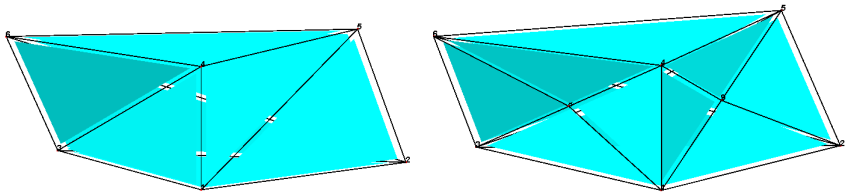


Figure A.2: Schönhardt's polyhedron , and its CDT with 3 Steiner points.

Fig. A.2). Fortunately, in carefully adding some points (called *Steiner points)*, the CDT of a PLS always exists ([Shewchuk, 2002], [Si, 2010]).

The following concepts and materials are reproduced from [Si, 2010] to understand CDT:

Piecewise Linear System (abbreviated as PLS) is a finite collection $\mathcal{X}$ of polyhedra with the following properties: (Fig. A.3)

1. if $P \in \mathcal{X}$, then all faces of P are in $\mathcal{X}$.
2. if $P, Q \in \mathcal{X}$, then $P \bigcap Q \subset \mathcal{X}$.
3. if $\dim(P \bigcap Q) = \dim(P), P \neq Q$ then $P \subset Q$, and $\dim(P) < \dim(Q)$.

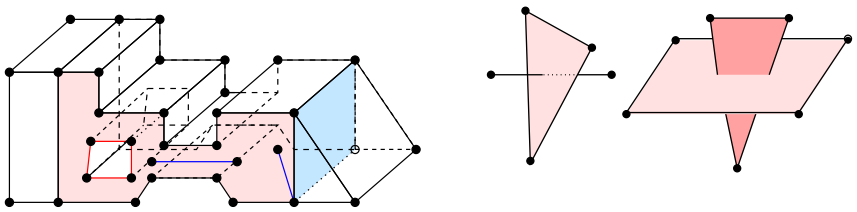


Figure A.3: The left image is a PLS, while the middle and the right one are not.

The dimension of a PLS $\mathcal{X}$, denoted as $\dim(\mathcal{X})$, is the largest dimension of its polyhedra. A subsystem of $\mathcal{X}$ is a subset of $\mathcal{X}$ which is also a PLS. The underlying space of $\mathcal{X}$ is $|\mathcal{X}| = \bigcup_{P \in \mathcal{X}} P$. The set of all vertices in $\mathcal{X}$ is denoted as $vert(\mathcal{X})$.

Triangulation of PLS Let $\mathcal{X}$ be a PLS in $\mathbb{R}^d$. A triangulation of $\mathcal{X}$ is a simplical complex $\mathcal{T}$ such that the underlying space of $\mathcal{T}$ equals the convex hull of the vertices of $\mathcal{X}$ and every polyhedron of $\mathcal{X}$ is represented by a subcomplex of $\mathcal{T}$. More formally:

1. $|\mathcal{T}| = conv(vert(\mathcal{X}))$ and
2. $\forall P \in \mathcal{X} \Rightarrow \exists \mathcal{K} \subset \mathcal{T}$ such that $|\mathcal{K}| = P$.

Visibility in $\mathbb{R}^3$: Two points $x, y \in \mathbb{R}^3$ are invisible to each other if the interior of the line segment xy intersects a polyhedron $P \in \mathcal{X}$ at a single point. Otherwise x and y are visible to each other (see an example in Fig. A.4).

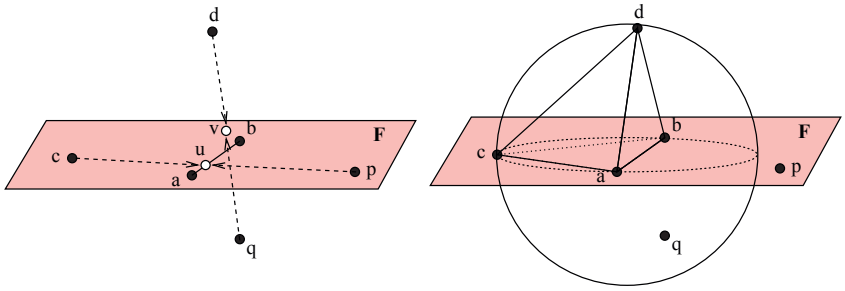


Figure A.4: Visibility and constrained Delaunay criterion. The shaded region is a facet F of a PLS $\mathcal{X}$, $a, b, c, p \in F$, $ab \in \mathcal{X}$. Left: d and q are invisible to each other because of F, c and p are invisible to each other because of segment ab. Right: a circumball of the tetrahedron $abcd$ contains q. $abcd$ is constrained Delaunay since q is not visible from its interior. The triangle $abc \subset F$ is constrained Delaunay since p is outside its diametric ball.

Constrained Delaunay simplex: Let S be a finite set of points and $\mathcal{X}$ be a PLS in $\mathbb{R}^3$ with $vert(\mathcal{X}) \subset S$. A simplex σ whose vertices are in S is constrained Delaunay if it is in one of the two cases:

1. There is a circumball B_σ of σ contains no vertices of S in its interior.

2. There exists $F \in \mathcal{X}$, such that $int(\sigma) \subset int(F)$. Let $K = S \bigcap aff(F)$, then no vertex of K contained in the interior of B_σ is visible from any point in $int(\sigma)$.

Constrained Delaunay tetrahedralization of $\mathcal{X}$ is defined as a tetrahedralization $\mathcal{T}$ of a PLS $\mathcal{X}$ such that every simplex of $\mathcal{X}$ is contrained Delaunay.

Hang Si [Si, 2010] presented his algorithm for CDT and proved that this algorithm terminates. His method can compute CDT for any PLS in $\mathbb{R}^3$, with Steiner points may be added in some segments of the PLS. He claimed that his method tended to add less Steiner points than [Shewchuk, 2002].

A.3 Graph cuts optimization

Given a finite directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V} = \{v_1, \ldots, v_n\}$ and edges $\mathcal{E}$ with non-negative weights (capacities) w_{pq}, and two special vertices, the source s and the sink t, an s-t-cut $\mathcal{C} = (\mathcal{S}, \mathcal{T})$ is a partition of $\mathcal{V}$ into two disjoints sets $\mathcal{S}$ and $\mathcal{T}$ such that $s \in \mathcal{S}$ and $t \in \mathcal{T}$ (Fig. A.5).

The cost of the cut is the sum of the capacities of all the edges going from $\mathcal{S}$ to $\mathcal{T}$:

$$c(\mathcal{S},\mathcal{T}) = \sum_{\substack{v_p\in\mathcal{S}\setminus\{s\}\\ v_q\in\mathcal{T}\setminus\{t\}}} w_{pq} + \sum_{v_p\in\mathcal{S}\setminus\{s\}} w_{pt} + \sum_{v_p\in\mathcal{T}\setminus\{t\}} w_{sp} \tag{A.1}$$

The minimum s-t-cut problem consists in finding a cut $\mathcal{C}$ with the smallest cost: the Ford-Fulkerson theorem [Ford and Fulkerson, 1962] states that this problem is equivalent to computing the maximum flow from the source s to the sink t and many classical algorithms exist to efficiently solve this problem.

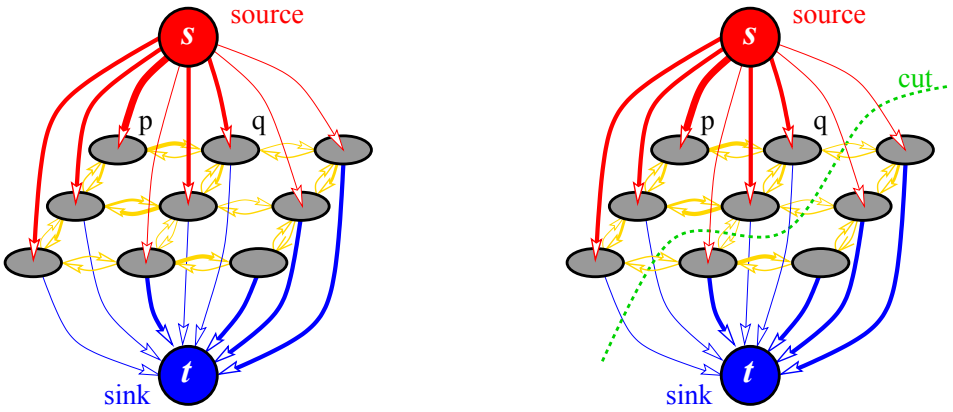


Figure A.5: A graph and a cut on it. Edge costs are reflected by thickness (image is reproduced from [Boykov and Kolmogorov, 2004].

Graph cuts optimization has been widely used in multi-view stereo and many other domains of computer vision [Boykov and Kolmogorov, 2004]. It is efficiently applied in segmentation, multi-view, optical flow, texturing, etc. There is a wide range of energy functions which can be cast as a graph cuts optimization [Kolmogorov and Zabih, 2004]. Graph cuts is also used to solve a multi-label problem [Boykov et al., 2001].

Graph cuts optimization can only handle a limited class of discrete energy. For more general energy functions and more performance, there are other methods such as belief propagation, tree re-weighted message passing [Wainwright et al., 2005], [Kolmogorov, 2006], linear programming [Komodakis et al., 2011], [Komodakis and Tziritas, 2007].

A.4 kd-tree

Binary space partitioning (BSP) is a method for recursively subdividing a space into convex sets by hyperplanes. This subdivision leads to a representation of the scene by means of a binary tree data structure (BSP tree).

kd-tree (short for *k-dimensional tree*) ([Bentley, 1975]) is a space partitioning data structure for organizing points in k-dimensional space. kd-tree is a special case

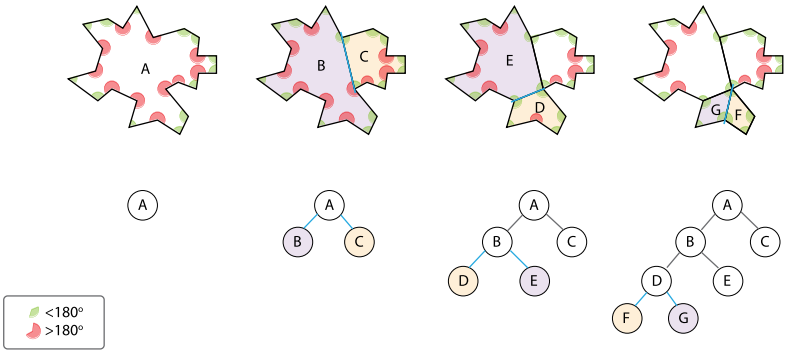


Figure A.6: A process of partitioning an irregular polygon into a series of convex ones by BSP (Image created by Jkwchui in wikipedia).

of BSP, when splitting hyperplanes are perpendicular to a dimensional axis. The hyperplane divides the space into two pars, known as subspaces. Points in each subspace represent a sub-tree.

Please note that is an informal definition of kd-tree. There are many ways to choose axis-aligned splitting tree. The nearest neighbor search (NN) usually need a balance kd-tree, then the splitting hyperplanes is chosen at the median of one coordinate of point set. In chapter 6, we use hyperplane in the middle of the longest axis of a node's bounding box to generate point clusters. It is optional to consider points are only in leaf nodes. The average complexity of kd-tree construction is around $O(n \log n)$.An example of kd-tree is shown in Fig. A.7.

A.5 kd-tree of volumetric objects

Instead of points, a kd-tree can contain other geometric like: spheres, segments, rectangles, triangles, etc. In this thesis, this kd-tree helps to detect quickly intersection of shapes. For example, we want to determine among n triangles in $\mathbb{R}^3$, all pairs of triangles that intersect each other. A brute force algorithm tests intersection of all 2 triangles with quadratic complexity $O(n^2)$. It's better to create a kd-tree structure in which every triangle is put in its leaf-nodes, where every intersected triangles are found in the same leaf-nodes.

To this end, the bounding box of the shapes is recursively split by hyperplanes perpendicular to an axis like normal kd-tree. However, each subspace will contain all shapes that belong or intersect that subspace, hence a shape may be contained in many nodes. The leaf node is the node that we can not divide it into 2 nodes that have the number of shapes strictly less than its number of shapes. By this construction, we ensure that any pair of intersected triangles will be found in at least

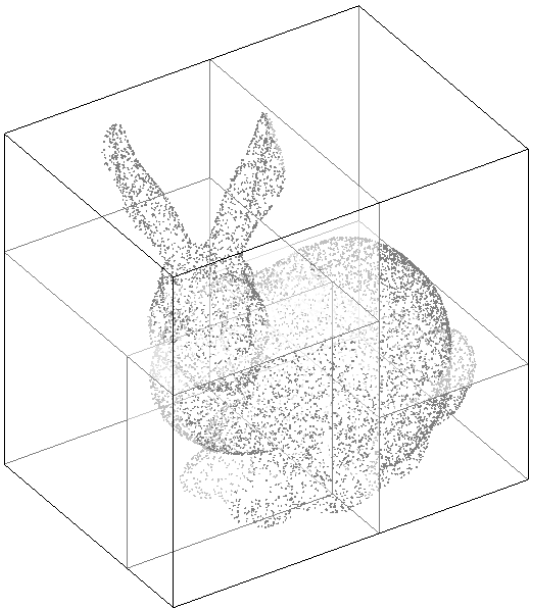


Figure A.7: Partition of a Stanford Bunny point set in 5 equal parts in using kd-tree. Each box represents a leaf node of the tree.

one leaf-nodes. In this thesis, for spheres, we consider all its center, for triangles, we consider all its vertices to build a kd-tree. For each node, we choose hyperplane at the medium of coordinate of all points in longest axis. It reduces the number of shapes in each node by roughly a half. While a pair of intersected shapes may be in many leaf-nodes, we can test it only one time by mark it as 'tested' so that we ignore it in the other nodes. The complexity of building the kd-tree is roughly $O(n \log n)$. The complexity of the intersection detection is a constant factor of the number of intersections. Then it is more efficient than brute force test if the number of intersected pairs is much smaller than n^2.

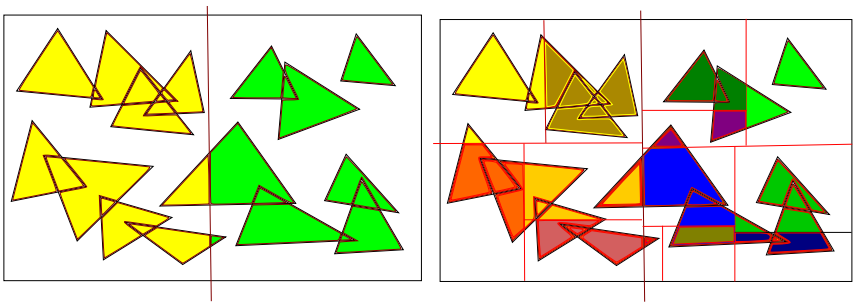


Figure A.8: kd-tree for triangles to determine the intersections. Left: the root is divided in 2 nodes by a hyperplane. Right: each cell represent leaf-trees containing triangles. Each color corresponds to a node. A triangle with many colors belongs to different nodes.

One special case of intersection using kd-tree is the determination of segments containing a set of Steiner points. The Constrained Delaunay Tetrahedralization of

a set of constrained triangles may require additional Steiner points in the edges of these triangles. While it's possible to modify the tetgen program so that it reports Steiner points associated with segments containing them, we prefer to use it as a black box and detect the segments later. The detection is done with a kd-tree of segments: when a node is divided in two child-nodes, we also distribute Steiner points in each node. We delete a node if it does not contain any Steiner points. At the end, in the leaf-nodes, we test for all Steiner points and all segments to determine which segments contain Steiner points. Again, it's much faster than a brute force approach, especially when the number of constrained segments are much more than the number of Steiner points.

Appendix B

Supplement results

We add the results of some image data-sets of CMP, Czech Technical University in Prague [1]. We test 3 data-sets: Head data-set, Detenice fountain data-set, and Tête de Plate Longe dataset. They are given as set of images without calibration camera. We then calibrate them with Bundler [Snavely et al., 2006], which might not calibrate all images of each data-set. Next, we build a set of pairs of images based on SfM point cloud, for the multi-view stereo, in which we may not use all calibrated images (Fig.B.1). The results are given in Fig. B.2, Fig. B.3 and Fig. B.4.

Figure B.1: Some images 3 calibrated datasets: Head (61 images of 6 MPixel, 126 pairs), Delenice Fountain(118 images of 3 MPixel , 118 pairs), and Tete (200 images of 5 Mpixel, 700 pairs).

[1] http://cmp.felk.cvut.cz/projects/is3d/Data.html

Figure B.2: Result of Head data-set with full resolution (mipmap 0: 6 MPixel). This data-set is very well calibrated, so that the reconstructed mesh is highly accurate. Last 2 images show the mesh with texture.

Figure B.3: Result of Detenice data-set with demi-resolution (mipmap 1: 750K Pixel). The result with full resolution does not improve the quality of the mesh. It might be due to the limited image quality (chromatic aberration, blur, etc) which leads to not accurate calibration and multi-view stereo. Last 2 images show the mesh with texture.

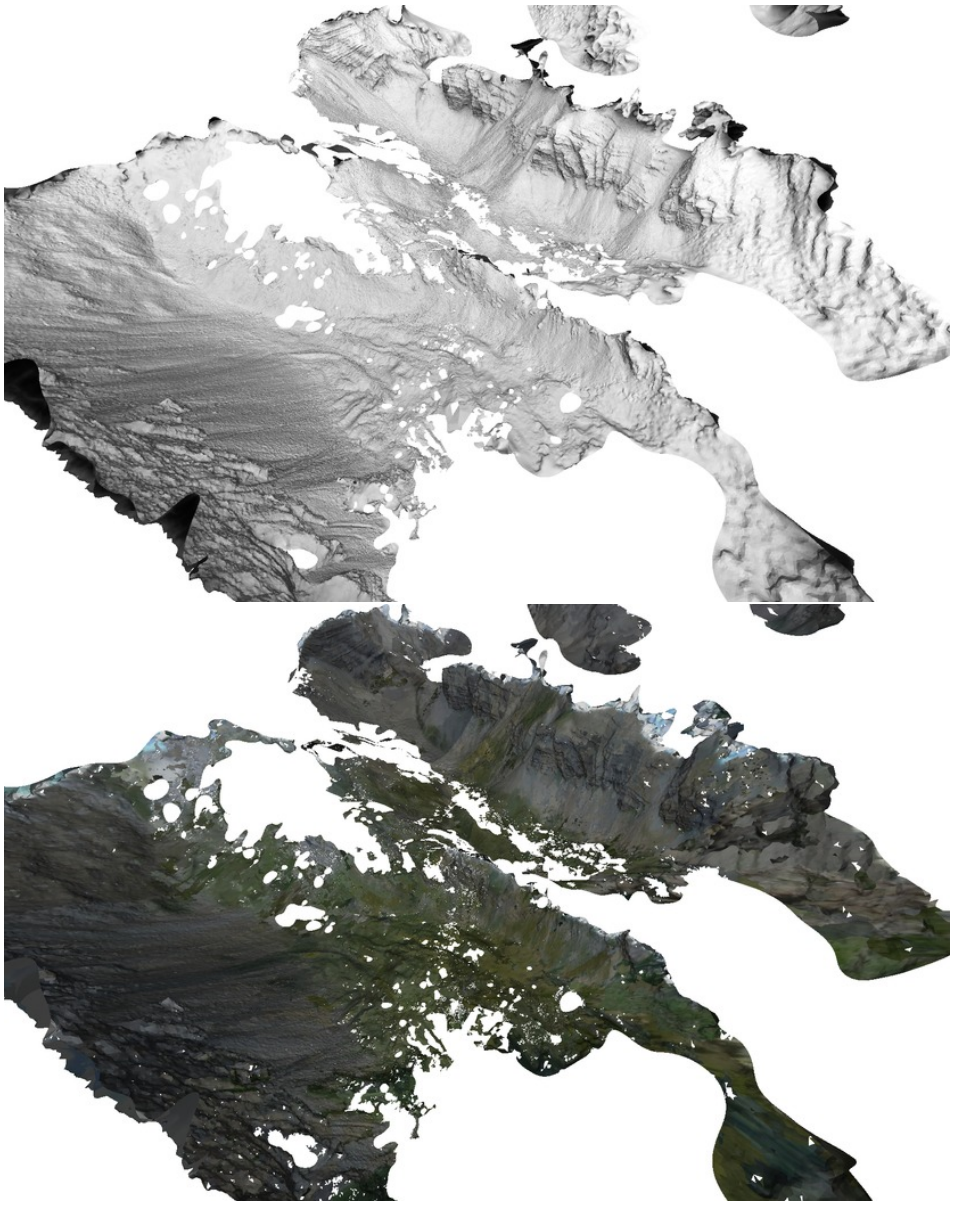


Figure B.4: Result of Tete data-set with demi-resolution (mipmap 1: 1.2M Pixel): mesh without and with texture. The result with full resolution does not visually improve the quality of the mesh.

Bibliography

Aganj, E., Pons, J.-P., and Keriven, R. (2009). Globally optimal spatio-temporal reconstruction from cluttered videos. In *Asian Conference on Computer Vision*. (Cited on page 32.)

Agarwal, S., Snavely, N., Simon, I., M.Seitz, S., and Szeliski, R. (2009). Building Rome in a day. In *International Conference on Computer Vision*. (Cited on page 30.)

Alegre, F. and Dellaert, F. (2004). A probabilistic approach to the semantic interpretation of building facades. In *Workshop on Vision Techniques Applied to the Rehabilitation of City Centres*. (Cited on page 33.)

Amenta, N., Bern, M., and Eppstein, D. (1998). The crust and the beta-skeleton: Combinatorial curve reconstruction. *Graphical Models and Image Processing*, 60(2):125–135. (Cited on page 74.)

Amenta, N., Choi, S., Dey, T. K., and Leekha, N. (2002). A simple algorithm for homeomorphic surface reconstruction. *International Journal of Computational Geometry and Applications*, 12(1–2):125–141. (Cited on page 74.)

Amenta, N., Choi, S., and Kolluri, R. (2001). The power crust. In *ACM Symposium on Solid Modeling and Applications*, pages 249–260. (Cited on page 58.)

Banno, A., Masuda, T., Oishi, T., and Ikeuchi, K. (2008). Flying laser range sensor for large-scale site-modeling and its applications in Bayon digital archival project. *International Journal of Computer Vision*, 78(2–3):207–222. (Cited on page 36.)

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18:509–517. (Cited on pages 39 and 118.)

Bernardini, F., Mittlement, J., Rushmeier, H., Silva, C., and Taubin, G. (1999). The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*. (Cited on pages 74 and 75.)

Boissonnat, J.-D. (1984). Geometric structures for three-dimensional shape representation. *ACM Transactions on Graphics*, 3:266–286. (Cited on page 74.)

Boissonnat, J.-D., Devillers, O., Teillaud, M., and Yvinec, M. (2000). Triangulations in CGAL. In *ACM Symposium on Computational Geometry*. (Cited on page 49.)

Boissonnat, J.-D. and Oudot, S. (2005). Provably good sampling and meshing of surfaces. *Graphical Models*, 67(5):405–451. (Cited on page 58.)

Bolitho, M., Kazhdan, M., Burns, R., and Hoppe, H. (2007). Multilevel streaming for out-of-core surface reconstruction. In *Proceedings of the fifth Eurographics symposium on Geometry processing*. (Cited on pages 30 and 75.)

Boykov, Y. and Kolmogorov, V. (2004). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137. (Cited on pages 49, 63 and 118.)

Boykov, Y. and Lempitsky, V. (2006). From photohulls to photoflux optimization. In *British Machine Vision Conference*, volume 3, pages 1149–1158. (Cited on page 28.)

Boykov, Y., Veksler, O., and Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:1222–1239. (Cited on page 118.)

Bradley, D., Boubekeur, T., and Heidrich, W. (2008). Accurate multi-view reconstruction using robust binocular stereo and surface meshing. In *IEEE Conference on Computer Vision and Pattern Recognition*. (Cited on pages 24 and 50.)

Broadhurst, A., Drummond, T. W., and Cipolla, R. (2001). A probabilistic framework for space carving. In *IEEE International Conference on Computer Vision*, pages 388–393. (Cited on pages 26, 28 and 73.)

Campbell, N. D. F., Vogiatzis, G., Hernández, C., and Cipolla, R. (2008). Using multiple hypotheses to improve depth-maps for multi-view stereo. In *European Conference on Computer Vision*, volume 5302 of *Lecture Notes in Computer Science*, pages 766–779. (Cited on pages 23, 25, 28 and 50.)

Cazals, F. and Giesen, J. (2006). *Effective Computational Geometry for Curves and Surfaces*, chapter Delaunay Triangulation Based Surface Reconstruction, pages 231–276. Springer. (Cited on pages 59 and 74.)

Cech, J. and Sara, R. (2007). Efficient sampling of disparity spacefor fast and accurate matching. In *IEEE Conference on Computer Vision and Pattern Recognition*. (Cited on pages 23 and 24.)

Chauve, A.-L., Labatut, P., and Pons, J.-P. (2010). Robust piecewise-planar 3d reconstruction and completion from large-scale unstructured point data. In *Con-*

ference on Computer Vision and Pattern Recognition (CVPR), San Fransisco. (Cited on page 33.)

Chew, L. P. (1989). Constrained delaunay triangulations. *Algorithmica*, 4. (Cited on page 115.)

Collins, R. (1996). A space-sweep approach to true multi-image matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 358–363. (Cited on page 22.)

Courchay, J., Pons, J.-P., Monasse, P., and Keriven, R. (2009). Dense and accurate spatio-temporal multi-view stereovision. In *Asian Conference on Computer Vision*. (Cited on page 32.)

Crandall, D., Owens, A., Snavely, N., and Huttenlocher, D. (2011). Discrete-continuous optimization for large-scale structure from motion. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, Colorado. (Cited on page 31.)

Curless, B. and Levoy, M. (1996). A volumetric method for building complex models from range images. In *ACM SIGGRAPH*, pages 303–312. (Cited on pages 25, 32, 58, 74 and 75.)

Delaunoy, A., Prados, E., Gargallo, P., Pons, J.-P., and Sturm, P. (2008). Minimizing the multi-view stereo reprojection error for triangular surface meshes. In *British Machine Vision Conference*. (Cited on pages 22, 24, 26, 28, 46 and 73.)

Dey, T. K. and Goswami, S. (2006). Provable surface reconstruction from noisy samples. *Computational Geometry: Theory and Applications*, 35(1):124–141. (Cited on page 74.)

Duan, Y., Yang, L., Qin, H., and Samaras, D. (2004). Shape reconstruction from 3D and 2D data using PDE-based deformable surfaces. In *European Conference on Computer Vision*, volume 3, pages 238–251. (Cited on pages 28 and 45.)

Eckstein, I., Pons, J.-P., Tong, Y., Kuo, C. C. J., and Desbrun, M. (2007). Generalized surface flows for mesh processing. In *Symposium on Geometry Processing*, pages 183–192. (Cited on page 46.)

Faugeras, O. and Keriven, R. (1998). Variational principles, surface evolution, PDE's, level set methods and the stereo problem. *IEEE Transactions on Image Processing*, 7(3):336–344. (Cited on pages 21, 23, 26, 28, 43, 44, 45 and 73.)

Ford, L. R. and Fulkerson, D. R. (1962). *Flows in Networks*. Princeton University Press. (Cited on page 118.)

Frahm, J.-M., Georgel, P., Gallup, D., Johnson, T., Raguram, R., Wu, C., Jen, Y.-H., Dumn, E., Clipp, B., Lazebnik, S., and Pollefeys, M. (2010). Building Rome on a cloudless day. In *European Conference on Computer Vision*. (Cited on pages 22, 29 and 30.)

Furukawa, Y., Curless, B., Seitz, S. M., and Szeliski, R. (2009). Manhattan-world stereo. In *IEEE Conference on Computer Vision and Pattern Recognition*. (Cited on page 33.)

Furukawa, Y., Curless, B., Seitz, S. M., and Szeliski, R. (2010). Towards Internet-scale multi-view stereo. In *IEEE Conference on Computer Vision and Pattern Recognition*. (Cited on pages 29, 30, 68 and 95.)

Furukawa, Y. and Ponce, J. (2006). Carved visual hulls for image-based modeling. In *European Conference on Computer Vision*, pages 564–577. (Cited on pages 26, 27 and 28.)

Furukawa, Y. and Ponce, J. (2007). Accurate, dense, and robust multi-view stereopsis. In *IEEE Conference on Computer Vision and Pattern Recognition*. (Cited on page 104.)

Furukawa, Y. and Ponce, J. (2008). Accurate, dense, and robust multi-view stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. (Cited on pages 21, 23, 25, 29, 30, 40, 50, 51, 53, 73, 75, 96, 104 and 112.)

Gargallo, P. (2008). *Contributions to the Bayesian Approach to Multi-View Stereo*. PhD thesis, INPG. (Cited on page 18.)

Gargallo, P., Prados, E., and Sturm, P. (2007). Minimizing the reprojection error in surface reconstruction from images. In *IEEE International Conference on Computer Vision*. (Cited on pages 22, 24 and 26.)

Gargallo, P. and Sturm, P. (2005). Bayesian 3D modeling from images using multiple depth maps. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 885–891. (Cited on pages 23, 25, 28 and 73.)

Goesele, M., Curless, B., and Seitz, S. M. (2006). Multi-view stereo revisited. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2402–2409. (Cited on pages 25, 28 and 73.)

Goesele, M., Snavely, N., Curless, B., Hoppe, H., and Seitz, S. M. (2007). Multi-view stereo for community photo collections. In *IEEE International Conference on Computer Vision*. (Cited on pages 23, 25, 28, 30, 40 and 73.)

Hartley, R. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press. (Cited on pages 3, 11, 18, 30, 31 and 38.)

Hernández, C. and Schmitt, F. (2004). Silhouette and stereo fusion for 3D object modeling. *Computer Vision and Image Understanding, special issue on 'Model-based and image-based 3D Scene Representation for Interactive Visualization'*, 96(3):367–392. (Cited on pages 22, 23, 26, 27, 28, 46, 50 and 73.)

Hernández, C., Vogiatzis, G., and Cipolla, R. (2007). Probabilistic visibility for multi-view stereo. In *IEEE Conference on Computer Vision and Pattern Recognition*. (Cited on pages 27, 28, 29, 42 and 73.)

Hirschmueller, H. and Scharstein, D. (2007). Evaluation of cost functions for stereo matching. In *IEEE Conference on Computer Vision and Pattern Recognition*. (Cited on pages 4 and 11.)

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W. (1992). Surface reconstruction from unorganized points. In *ACM SIGGRAPH*, pages 71–78. (Cited on page 74.)

Hornung, A. and Kobbelt, L. (2006a). Hierarchical volumetric multi-view stereo reconstruction of manifold surfaces based on dual graph embedding. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 503–510. (Cited on pages 26, 27 and 28.)

Hornung, A. and Kobbelt, L. (2006b). Robust and efficient photo-consistency estimation for volumetric 3D reconstruction. In *European Conference on Computer Vision*. (Cited on pages 21, 26 and 27.)

Huang, X., Fu, H., Au, O. K.-C., and Tai, C.-L. (2007). Optimal boundaries for poisson mesh merging. In *Proceedings of the 2007 ACM symposium on Solid and physical modeling*, SPM '07, pages 35–40, New York, NY, USA. ACM. (Cited on page 58.)

Jancosek, M. and Pajdla, T. (2009). Segmentation based multi-view stereo. In *Computer Vision Winter Workshop*. (Cited on page 53.)

Jancosek, M. and Pajdla, T. (2011). Multi-view reconstruction preserving weakly-supported surface. In *IEEE Conference on Computer Vision and Pattern Recognition*. (Cited on pages 25 and 73.)

Jancosek, M., Shekhovtsov, A., and Pajdla, T. (2009). Scalable multi-view stereo. In *The 2009 IEEE International Workshop on 3-D Digital Imaging and Modeling*. (Cited on pages 29, 51, 53, 68 and 96.)

Jin, H., Soatto, S., and Yezzi, A. J. (2005). Multi-view stereo reconstruction of dense shape and complex appearance. *International Journal of Computer Vision*, 63(3):175–189. (Cited on pages 23, 26, 28 and 45.)

Kazhdan, M., Bolitho, M., and Hoppe, H. (2006). Poisson surface reconstruction. In *Symposium on Geometry Processing*, pages 61–70. (Cited on pages 25, 29, 58 and 75.)

Kim, S. J., Gallup, D., Frahm, J.-M., Akbarzadeh, A., Yang, Q., Yang, R., Nister, D., and Pollefeys, M. (2007). Gain adaptive real-time stereo streaming. In *International Conference on Computer Vision Systems*. (Cited on page 22.)

Kobbelt, L., Campagna, S., Vorsatz, J., and Seidel, H.-P. (1998). Interactive multi-resolution modeling on arbitrary meshes. In *International Conference on Computer Graphics and Interactive Techniques*, pages 105–114. (Cited on page 48.)

Kolev, K., Klodt, M., Brox, T., and Cremers, D. (2009). Continuous global optimization in multiview 3D reconstruction. *International Journal of Computer Vision*. (Cited on pages 28 and 50.)

Kolmogorov, V. (2006). Convergent tree-reweighted message passing for energy minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. (Cited on page 118.)

Kolmogorov, V. and Zabih, R. (2001). Computing visual correspondence with occlusions via graph cuts. In *IEEE International Conference on Computer Vision*, volume 2, pages 508–? (Cited on page 73.)

Kolmogorov, V. and Zabih, R. (2002). Multi-camera scene reconstruction via graph cuts. In *European Conference on Computer Vision*, volume 3, pages 82–96. (Cited on pages 23, 25, 28 and 73.)

Kolmogorov, V. and Zabih, R. (2004). What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–159. (Cited on page 118.)

Komodakis, N., Paragios, N., and Tziritas, G. (2011). Mrf energy minimization & beyond via dual decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. (Cited on page 118.)

Komodakis, N. and Tziritas, G. (2007). Approximate labeling via graph cuts based on linear programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. (Cited on page 118.)

Kutulakos, K. N. and Seitz, S. M. (2000). A theory of shape by space carving. *International Journal of Computer Vision*, 38(3):199–218. (Cited on pages 26 and 28.)

Labatut, P., Pons, J.-P., and Keriven, R. (2007). Efficient multi-view reconstruction of large-scale scenes using interest points, Delaunay triangulation and graph cuts. In *IEEE International Conference on Computer Vision*. (Cited on pages 25, 36, 42, 59 and 61.)

Labatut, P., Pons, J.-P., and Keriven, R. (2009a). Hierarchical shape-based surface reconstruction for dense multi-view stereo. In *The 2009 IEEE International Workshop on 3-D Digital Imaging and Modeling*. (Cited on page 33.)

Labatut, P., Pons, J.-P., and Keriven, R. (2009b). Robust and efficient surface reconstruction from range data. *Computer Graphics Forum*, 28:2275–2290. (Cited on pages 25, 36, 42, 59, 61, 62, 74 and 75.)

Lafarge, F., Keriven, R., Brédif, M., and Vu, H. H. (2010). Hybrid multi-view reconstruction by jump-diffusion. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, San Fransisco. (Cited on pages 33 and 53.)

Laurentini, A. (1994). The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162. (Cited on page 27.)

Lee, D. and Lin, A. (1986). Generalized delaunay triangulation for planar graphs. *Discrete & Computational Geometry*, 1:201–217. (Cited on page 115.)

Lempitsky, V. and Boykov, Y. (2007). Global optimization for shape fitting. In *IEEE Conference on Computer Vision and Pattern Recognition*. (Cited on page 42.)

Lempitsky, V., Boykov, Y., and Ivanov, D. (2006). Oriented visibility for multiview reconstruction. In *European Conference on Computer Vision*, volume 3, pages 226–238. (Cited on page 28.)

Levoy, M., Rusinkiewicz, S., Curless, B., Ginzton, M., Ginsberg, J., Pulli, K., Koller, D., Anderson, S., Shade, J., Pereira, L., Davis, J., and Fulk, D. (2000). The digital michelangelo project: 3d scanning of large statues. In *ACM SIGGRAPH*, pages 131–144. (Cited on page 75.)

Lhuillier, M. and Quan, L. (2005). A quasi-dense approach to surface reconstruction from uncalibrated images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):418–433. (Cited on pages 23, 26, 28 and 45.)

Lovi (2010). Incremental free-space carving for real-time 3d reconstruction. In *Fifth International Symposium on 3D Data Processing Visualization and Transmission(3DPVT)*. (Cited on pages 25, 73, 74 and 91.)

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110. (Cited on pages 4 and 11.)

Merrell, P., Akbarzadeh, A., Wang, L., Mordohai, P., Frahm, J.-M., Yang, R., Nistér, D., and Pollefeys, M. (2007). Real-time visibility-based fusion of depth maps. In *IEEE International Conference on Computer Vision*. (Cited on pages 22, 24, 28 and 73.)

M uller, P., Wonka, P., Haegler, S., Ulmer, A., and Gool, L. V. (2006). Procedural modeling of buildings. 25(3):614–623. (Cited on page 33.)

Pajarola, R. (2005). Stream-processing points. In *In Proceedings of the Conference on Visualization*. (Cited on page 75.)

Pan, Q., Reitmayr, G., and Drummond, T. (2009). Proforma: Probabilistic feature-based on-line rapid model acquisition. In *Proc. 20th British Machine Vision Conference (BMVC)*, London. (Cited on pages 25, 73, 74 and 91.)

Paragios, N., Chen, Y., and Faugeras, O. (2005). *The Handbook of Mathematical Models in Computer Vision*. Springer. (Cited on page 30.)

Pons, J.-P. (2005). *Contributions Méthodologiques et Appliquées à la Méthode des Modèles Déformables*. PhD thesis, École Nationale des Ponts et Chaussées. (Cited on page 18.)

Pons, J.-P., Keriven, R., and Faugeras, O. (2007). Multi-view stereo reconstruction and scene flow estimation with a global image-based matching score. *International Journal of Computer Vision*, 72(2):179–193. (Cited on pages 22, 24, 26, 28, 36, 44, 45, 46, 47, 50 and 73.)

Salman, N. and Yvinec, M. (2009). Surface reconstruction from multi-view stereo. In *The 2009 International Workshop on Representation and Modeling of Large-scale 3D Environments*. (Cited on pages 29, 51 and 53.)

Scharstein, D. and Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1–2–3):7–42. (Cited on pages 4, 12 and 24.)

Scheidegger, C. E., Fleishman, S., and Silva, C. T. (2005). Triangulating point set surfaces with bounded error. In *Proceedings of the third Eurographics symposium*

on Geometry processing, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association. (Cited on page 58.)

Seitz, S. M., Curless, B., Diebel, J., Scharstein, D., and Szeliski, R. (2006). A comparison and evaluation of multi-view stereo reconstruction algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 519–526. (Cited on pages 17, 20, 21, 27, 28 and 50.)

Seitz, S. M. and Dyer, C. R. (1999). Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision*, 35(2):151–173. (Cited on pages 21 and 28.)

Sharf, A., Blumenkrants, M., Shamir, A., and Cohen-Or, D. (2006). Snappaste: an interactive technique for easy mesh composition. *The Visual Computer*, 22:835–844. 10.1007/s00371-006-0068-5. (Cited on page 58.)

Shen, C., O'Brien, J. F., and Shewchuk, J. R. (2004). Interpolating and approximating implicit surfaces from polygon soup. In *ACM SIGGRAPH*, pages 896–904. (Cited on page 58.)

Shewchuk, J. (2002). Constrained delaunay tetrahedralizations and provably good boundary recovery. In *Eleventh International Meshing Roundtable (Ithaca, New York)*. (Cited on pages 116 and 117.)

Si, H. (2009). A quality tetrahedral mesh generator and a 3d delaunay triangulator. (Cited on page 61.)

Si, H. (2010). Constrained delaunay tetrahedral mesh generation and refinement. *Finite Elements in Analysis and Design*, 46(1-2):33–46. (Cited on pages 59, 116 and 117.)

Sinha, S. N., Mordohai, P., and Pollefeys, M. (2007). Multi-view stereo via graph cuts on the dual of an adaptive tetrahedral mesh. In *IEEE International Conference on Computer Vision*. (Cited on pages 21, 23, 26, 27 and 28.)

Sinha, S. N. and Pollefeys, M. (2005). Multi-view reconstruction using photo-consistency and exact silhouette constraints: A maximum-flow formulation. In *IEEE International Conference on Computer Vision*, pages 349–356. (Cited on page 28.)

Slabaugh, G. and Unal, G. (2005). Active polyhedron: Surface evolution theory applied to deformable meshes. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 84–91. (Cited on page 46.)

Snavely, N., Seitz, S. M., and Szeliski, R. (2006). Photo tourism: Exploring photo collections in 3D. In *ACM SIGGRAPH*. (Cited on pages 112 and 123.)

Snavely, N., Seitz, S. M., and Szeliski, R. (2008a). Modeling the world from internet photo collections. *International Journal of Computer Vision*, 80(2):189–210. (Cited on pages 29 and 31.)

Snavely, N., Seitz, S. M., and Szeliski, R. (2008b). Skeletal sets for efficient structure from motion. In *Proc. Computer Vision and Pattern Recognition*. (Cited on page 29.)

Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rossl, C., and Seidel, H.-P. (2004). Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, SGP '04, pages 175–184, New York, NY, USA. ACM. (Cited on page 58.)

Starck, J., Miller, G., and Hilton, A. (2006). Volumetric stereo with silhouette and feature constraints. In *British Machine Vision Conference*, volume 3, pages 1189–1198. (Cited on pages 21 and 27.)

Strecha, C., Fransens, R., and Gool, L. V. (2004). Wide-baseline stereo from multiple views: A probabilistic account. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 552–559. (Cited on pages 23, 25, 28, 53 and 73.)

Strecha, C., Fransens, R., and Gool, L. V. (2006). Combined depth and outlier estimation in multi-view stereo. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2394–2401. (Cited on pages 25, 28, 53 and 73.)

Strecha, C., Tuytelaars, T., and Gool, L. V. (2003). Dense matching of multiple wide-baseline views. In *IEEE International Conference on Computer Vision*, volume 2, pages 1194–1201. (Cited on pages 23, 28 and 73.)

Strecha, C., von Hansen, W., Gool, L. V., Fua, P., and Thoennessen, U. (2008). On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *IEEE Conference on Computer Vision and Pattern Recognition*. (Cited on pages 29, 39, 49 and 51.)

Sun, J., Li, Y., Kang, S. B., and Shum, H.-Y. (2005). Symmetric stereo matching for occlusion handling. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 399–406. (Cited on page 73.)

Szeliski, R. (1999). Prediction error as a quality metric for motion and stereo. In *Proceedings of the International Conference on Computer Vision*. (Cited on page 22.)

Szeliski, R. (2010). *Computer Vision, Algorithms and Application*. Springer. (Cited on pages 30 and 32.)

Taubin, G. (1995). A signal processing approach to fair surface design. In *ACM SIGGRAPH*, pages 351–358. (Cited on page 48.)

Teboul, O., Kokkinos, I., Simon, L., Koutsourakis, P., , and Paragios, N. (2011). Robust piecewise-planar 3d reconstruction and completion from large-scale unstructured point data. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, Colorado. (Cited on page 33.)

Tobor, I., Reuter, P., and Schlick, C. (2004). Multi-scale reconstruction of implicit surfaces with attributes from large unorganized point sets. In *Shape Modeling and Applications, International Conference on*, volume 0, pages 19–30, Los Alamitos, CA, USA. IEEE Computer Society. (Cited on pages 76 and 78.)

Tola, E., Lepetit, V., , and Fua, P. (2010). Daisy: an efficient dense descriptor applied to wide baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5):815–830. (Cited on pages 4 and 11.)

Tran, S. and Davis, L. (2006). 3D surface reconstruction using graph cuts with surface constraints. In *European Conference on Computer Vision*, volume 2, pages 219–231. (Cited on pages 21, 26, 27 and 28.)

Triggs, B., McLauchlan, P., Hartley3, R., and Fitzgibbon, A. (1999). Bundle adjustment - a modern synthesis. In *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, ICCV '99. (Cited on page 31.)

Turk, G. and Levoy, M. (1994). Zippered polygon meshes from range images. In *ACM SIGGRAPH*. (Cited on page 58.)

Tylecek, R. and Sara, R. (2009). Depth map fusion with camera position refinement. In *Computer Vision Winter Workshop*. (Cited on pages 23, 25, 51, 53, 73 and 112.)

Tylecek, R. and Sara, R. (2010). Refinement of surface mesh for accurate multi-view reconstruction. *The International Journal of Virtual Reality, 2010*. (Cited on pages 24, 25, 26, 29, 51, 53 and 75.)

Vogiatzis, G., Hernández, C., Torr, P. H. S., and Cipolla, R. (2007). Multi-view stereo via volumetric graph-cuts and occlusion robust photo-consistency. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2241–2246. (Cited on pages 21, 26 and 27.)

Vogiatzis, G., Torr, P. H. S., and Cipolla, R. (2005). Multi-view stereo via volumetric graph-cuts. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 391–398. (Cited on pages 21, 26, 27 and 28.)

Vu, H. H., Keriven, R., Labatut, P., and Pons, J.-P. (2009). Towards high-resolution large-scale multi-view stereo. In *IEEE Conference on Computer Vision and Pattern Recognition*. (Cited on pages 29 and 51.)

Vu, H. H., Labatut, P., Keriven, R., and Pons, J.-P. (2011). High accuracy and visibility-consistent dense multi-view stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. (Cited on pages 24, 26 and 29.)

Wainwright, M. J., Jaakkola, T. S., and Willsky, A. S. (2005). Map estimation via agreement on trees: Message-passing and linear programming. *IEEE Transactions on Information Theory*. (Cited on page 118.)

Wan, M., Wang, Y., and Wang, D. (2010). Variational surface reconstruction based on delaunay triangulation and graph cut. *International Journal for Numerical Methods in Engineering*. (Cited on page 59.)

Wojtan, C., Thürey, N., Gross, M., and Turk, G. (2009). Deforming meshes that split and merge. *ACM Trans. Graph.*, 28:76:1–76:10. (Cited on page 58.)

Yang, R. and Pollefeys, M. (2003). Multi-resolution real-time stereo on commodity graphics hardware. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 211–220. (Cited on page 38.)

Yang, R., Pollefeys, M., and Welch, G. (2003). Dealing with textureless regions and specular highlights: A progressive space carving scheme using a novel photo-consistency measure. In *IEEE International Conference on Computer Vision*, volume 1, pages 576–584. (Cited on pages 26 and 28.)

Yu, T., Ahuja, N., and Chen, W.-C. (2006). SDG cut: 3D reconstruction of non-lambertian objects using graph cuts on surface distance grid. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2. (Cited on pages 26 and 27.)

Zach, C., Pock, T., and Bischof, H. (2007). A globally optimal algorithm for robust TV-L1 range image integration. In *IEEE International Conference on Computer Vision*. (Cited on pages 22, 24, 25 and 28.)

Zaharescu, A., Boyer, E., and Horaud, R. (2011). Topology-adaptive mesh deformation for surface evolution, morphing, and multiview reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. (Cited on page 58.)

Zaharescu, A., Boyer, E., and Horaud, R. P. (2007). TransforMesh: a topology-adaptive mesh-based approach to surface evolution. In *Asian Conference on Computer Vision*, pages 166–175. (Cited on pages 50, 53 and 58.)

Zaharescu, A., Cagniart, C., Ilic, S., Boyer, E., and Horaud, R. P. (2008). Camera clustering for multi-resolution 3-D surface reconstruction. In *ECCV 2008 Workshop on Multi Camera and Multi-modal Sensor Fusion Algorithms and Applications*. (Cited on page 95.)

Zhao, H. K., Osher, S., and Fedkiw, R. (2001). Fast surface reconstruction using the level set method. In *IEEE Workshop on Variational and Level Set Methods in Computer Vision*, pages 194–201. (Cited on pages 23 and 26.)