IMAGE-BASED BUILDING MODELING

by

JIANXIONG XIAO

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfillment of the Requirements for
the Degree of Master of Philosophy
in Computer Science

August 2009, Hong Kong

Authorization

I hereby declare that I am the sole author of the thesis.

I authorize the Hong Kong University of Science and Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the Hong Kong University of Science and Technology to reproduce the thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

JIANXIONG XIAO

IMAGE-BASED BUILDING MODELING

by

JIANXIONG XIAO

This is to certify that I have examined the above M.Phil. thesis
and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by
the thesis examination committee have been made.

PROF. LONG QUAN, THESIS SUPERVISOR

PROF. MOUNIR HAMDI, HEAD OF DEPARTMENT

Department of Computer Science and Engineering
20 August 2009

ACKNOWLEDGMENTS

First and foremost, this thesis is not possible without my thesis advisor, Prof. Long Quan. He has supported me with generous and wise guidance, great encouragement and many fruitful discussions.

Many thanks to my thesis defense committee members, Prof. Dit-Yan Yeung and Prof. Chiew-Lan Tai. Prof. Yeung is a great mentor and friend to me personally. I am always encouraged by his rigorous attitude towards research and technical writing. My study in HKUST is impossible without the help of Prof. Tai. She is always very kind to offer useful information that greatly affects my academic career. I have also learned a lot about teaching when I was a TA for her course.

During my study in HKUST, I also have learned a lot from courses lectured by other faculty members: Prof. Pedro V. Sander, Prof. Jogesh K. Muppala, Prof. Siu-Wing Cheng, Prof. Chi-Keung Tang, Dr. Amy Chi, Prof. Albert C.S. Chung, Prof. Brian Kan-Wing Mak, Prof. Nevin L. Zhang, Prof. Philip Chi-Wing Fu, Prof. James Tin-Yau Kwok, Prof. Qiang Yang, Prof. Mordecai J. Golin, Prof. Bo Li, Prof. Frederick H. Lochovsky, Prof. Mounir Hamdi, Prof. Kin-Yin Li, Prof. Kani Chen, Prof. Mo Mu, and Dr. David Rossiter.

I want to thank other researchers with whom I worked together during my thesis: Prof. Ping Tan, Dr. Jingdong Wang, Lu Yuan, Tian Fang, Peng Zhao and Honghui Zhang. Dr. Yichen Wei and Prof. Gang Zeng also offered great help for my research. My research has kindly been supported by Research Grants Council (RGC) of Hong Kong and the National Natural Science Foundation of China (NSFC) under the project N-HKUST602/05, 619005, 619006, and 619107.

I am thankful to my labmates in Vision and Graphics Laboratory, and my friends for the friendship and help of my daily life. I am also grateful to the help from Ms. Vanessa Y.C. Kwok, Ms. Connie W.C. Lau, and Ms. Brenda Y.L. Tam during my study.

This thesis is impossible without the love from my parents and sister. My gratitude to them is beyond what I can express in words. Last but not least, I want to thank my wife for her love and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

IMAGE-BASED BUILDING MODELING

by

JIANXIONG XIAO

Department of Computer Science and Engineering

The Hong Kong University of Science and Technology

ABSTRACT

Image-based modeling is the process of converting 2D images of the real world into digital 3D models in computer. Among myriad kinds of objects in the world, man-made buildings are of special importance, since there are a large number of potential applications. This challenging problem has been studied by both academic research and commercial industrial communities. However, all the existing approaches still need plenty of human efforts in order to produce satisfactory results. This thesis presents a state-of-the-art automatic approach that only requires minimal human efforts for image-based building modeling to achieve visual pleasing results. This image-based approach has three steps: reconstruction, segmentation and modeling. The first step is to reconstruct a 3D point cloud from the 2D images. The second step is to segment the 3D point cloud and the 2D images, where each group represents an object or a kind of objects for modeling. The final step is to model each object by structure analysis and regularization. This three-step approach is remarkably robust, because it clearly divides the work into subproblems properly, and conquers each subproblem with strategies according to different objectives to be achieved in each stage. While this approach is also suitable for general image-based modeling of any object, specifical focus is on man-made buildings, where Manhattan-world property presents frequently. This thesis concludes with demonstration of the proposed approach for building modeling in several cities, including Pittsburgh, Minneapolis, Chapel Hill in the United States, and Guangzhou in China.

CHAPTER 1

INTRODUCTION

1.1 Motivation

Image-based modeling refers to the process of generating a three-dimensional digital model from a set of two-dimensional images of a scene, i.e. reconstructing existing models from photographs, and extracting texture maps directly from photographs to produce photo-realistic scenes of these meshes as it is in reality, for example, create 3D models of a town. Image-based modeling technology can be used for modeling scenes or buildings for architectural purposes, for archaeology databases, for calculations and 3D surveys from photographs, for creating 3D virtual visits and fast creation of photo-realistic backdrops, for reverse engineering, for human modeling, for industrial design, for animation and special effects, for simple creation of photo-realistic objects for video games, for forensics and plant engineering. With image-based modeling technology, we can use your computer offline to measure 3D distances and angles of real scenes: objects of any size, buildings, landscapes, far distances, inaccessible areas, crime scenes, etc. Furthermore, mechanical, civil and chemical engineers can also use image-based modeling for a diverse set of measuring and dimensioning tasks.

Especially, image-based modeling for buildings allows us to model of scenes or buildings "as built", create 3D virtual visits, simulate and prototype projects in their final environment, calculate scenes and 3D surveys from photographs, and match the camera perspective for overlay of 3D over 2D pictures. There are many applications for image-based building modeling, such as architectural planning, archaeological reconstructions, 3D earth maps, virtual reality, cultural heritage preservation, military mission simulation, driving training, and cinematic special effects for movies.

Therefore, a convenient image-based building modeling technology is the key for 3D modeling and measurement throughout many industry sectors such as architecture and preservation, archaeology and anthropology, film, video and animation, accident reconstruction.

Image-based building modeling is especially useful for architectural, preservation, conservation and cultural resource management applications, such as documenting and measuring older buildings and structures for conservation and preservation, generating 3D

models for visualization and view studies, creating elevation drawings of existing structures and rectified photographs of façades from photo projects, producing photo-textured 3D models for realistic walk-bys, surveying existing structures and objects, etc.

Image-based building modeling is a powerful tool for creating 3D models for animation and multimedia applications and is used extensively in animation, film and video, and web site design. We can build 3D models to use in animation and rendering programs, model objects for Computer Based Training, measure or model sets and locations, perform perspective matching to synchronize a CG camera to a real photo, export realistic texture maps from original photographs, create life-like photo-textured models with low polygon counts, etc.

Accident reconstruction firms, police forces, and forensic teams can also use photographs or video images taken at the scene, for image-based building modeling and reconstruction, for diagrams and maps of the scene creation, for measurements of distance, crush, and placement, for ortho-photos generation of skid marks and other surfaces, in order to make accurate measurements and maps quickly and easily and create archives of crime scene evidence for analysis at a later date.

1.2 History and the state-of-the-art

Image-based modeling is a subset of techniques under the more general photogrammetry, in which geometric properties about objects are determined from photographic images. Therefore, technically, photogrammetry means image-based geometric properties recovery. Since geometric properties recovery is almost equivalent to geometric modeling, the term "image-based modeling" is sometimes called "photogrammetry". Like other seemingly "modern invention" such as ray-tracing (1960, not 1984) and subdivision surfaces (1978, not 1998), photogrammetry is actually an old idea revived for modern use in a new application: digital 3D modeling. A long history can be traced back hundreds of years ago [11, 12, 13, 14].

As early as 1851, Aimé Laussedat, a military engineer in France, had begun experiments to use images for topographic mapping purposes. In the early period he worked with hand drawn images, acquired with the help of an optical tool for perspective drawing. Later he started to apply photographs, and in 1859 the prototype of a topographic camera was built to his specifications. It is therefore generally accepted that Laussedat is the "father of photogrammetry".

However, the origin of the term "photogrammetry" came from Albrecht Meydenbauer, who began his investigations into the use of photographs in measurement without any

(a) Aimé Laussedat (1819–1907)

(b) Albrecht Meydenbauer (1834–1921)

(c) Édouard-Gaston Deville (1849–1924)

Figure 1.1: **Father of photogrammetry.**

knowledge about the activities of Aimé Laussedat. In 1867, Dr. Meydenbauer published a paper on this subject in Weekly Journal of the Association of Architects in Berlin with the title "Die Photometrographie", in which the first use of the word photogrammetry appears. Dr. Otto Kersten, a geographical explorer, proposed to replace it by the much more convenient term "Photogrammetrie", and Meydenbauer accepted.

In 1885, Édouard-Gaston Deville, Surveyor General of Canada, experimented with mapping methods developed by Aimé Laussedat to survey in the Rocky Mountains. Édouard-Gaston used Laussedat's principle of elevated photography, and refined a technique of creating large-scale maps from these photographs. He designed a rugged, lightweight field camera that could be carried long distances. In 1886, his camera was used for the first time in the Rockies. Édouard-Gaston introduced his innovative mapping technique at the Chicago World Fair in 1893, later promoting it through pamphlets and a comprehensive textbook. This same basic process has been in continuous use for many years. Indeed, the Institute of Photogrammetry and Remote Sensing in Finland has been teaching photogrammetry as a formal course since 1946.

In 1996, Paul Debevec, Camillo Taylor and Jitendra Malik from UC Berkeley reintroduced the photogrammetry as the modern image-based modeling technique for computer graphics [15]. After that, many scientific researchers and industrial engineers realized the great importance of image-based modeling, especially for building, and many products are introduced into the markets.

1.2.1 Research projects

As mentioned above, under supervision of Jitendra Malik, Paul Debevec and Camillo Taylor reintroduced photogrammetry as image-based modeling in 1996 [16, 15]. Their system, called Façade, consists of three main component:

- Photogrammetric Modeling: A method for interactively recovering 3D models and camera positions from photographs;
- View-Dependent Texture Mapping: A method for turning a 3D model and a set of photographs from known positions into renderings;
- Model-Based Stereo: A method of refining an approximate geometric model of a scene to conform to its actual appearance in a set of photographs.

Façade itself remains an unreleased research prototype, but much of its functionality is now available in commercial products. Façade was a source of inspiration for MetaCreations' product Canoma written by Robert Seidl and Tilman Reinhardt. In August 2000 Adobe Systems acquired Canoma and may integrate the technology into a new project to be released at a future date. Along with computer vision research led by Olivier Faugeras at INRIA, Façade was also a source of inspiration for RealViz's ImageModeler image-based modeling and rendering package, recently acquired by Autodesk.

While Debevec *et al.* focus on using manual effort for modeling, Seth Teller in MIT led a project, MIT City Scanning Project [17, 18, 19, 20, 21, 22, 23], to focus on fully-automated model acquisition in urban areas. Their approach was organized into: sensing, refinement, and model extraction. They used a pose camera to acquire pose imagery for spherical mosaics construction. Then, pose-imagery is refined by translational alignment and rotational alignment. Coarse structure is extracted for consensus texture estimation. Finally, the detailed relief geometry on each main plane is reconstructed.

Funded by Army Research Office (ARO) of the US government under Multidisciplinary University Research Initiative (MURI) program, the project "Next Generation 4-D Distributed Modeling and Visualization of Battlefield" was a collaboration among research groups led by Avideh Zakhor from UC Berkeley, Ulrich Neumann and Suya You from USC, William Ribarsky from Georgia Tech and UNC Charlotte, Pramod Varshney from Syracuse, and Suresh Lodha from UC Santa Cruz.

Zakhor *et al.* have proposed a system that contains three steps [24, 25, 26, 27, 28, 29, 30, 31]: airborne modeling, ground-based modeling and model fusion. The system first generates 3D models of rooftops and terrain shape from airborne laser scans and

photos, by processing airborne laser scans, reconstructing surface geometry, and texture mapping. Then, it generates 3D models of façades and street scenery as seen from street level, by ground-based data acquisition, position estimation using scan matching and Monte-Carlo localization, automated segmentation, mesh generation, simplification and texturing. Finally, the system merges ground-based and airborne model to one single model, usable for both walk- and fly-thrus. Zakhor *et al.* have developed a scheme which can automatically register aerial imagery onto 3D geometry models in a matter of minutes instead of hours. They start with a coarse GPS/INS readout and then refine the pose by applying a series of algorithms including: (a) vanishing point detection; (b) 2D corner detection and correspondence between 3D model and 2D imagery; (c) Hough transform to prune possible matches; (d) generalized RANSAC algorithm to find in-lier matches; (e) Bundle adjustment to compute the 3D camera pose. By taking advantage of the parallelism and orthogonality inherently present in man-made structures, their system is able to apply well-justified algorithms to provide a fast and truly automated camera registration solution for texture mapping.

To improve result quality of the fully automatic reconstruction, Neumann *et al.* proposed a semi-automatic system [32, 33, 34, 35, 36, 37, 38] to enable the user-guided interaction. A semi-automated primitive fitting technique is proposed for surface extraction to build models from LiDAR data. Primitives are able to fit to image in a few clicks using primitive constrained outline extraction and user-guided interactive extraction. They propose a new algorithm "Vanishing Hull" for robust vanishing point estimation to ensure edges are vertical and horizontal for uncalibrated texture images rectification. To remove texture occlusion, a smart removal and filling tool is used to find regions of "best match" between images for seamless transitions. They also proposed a technique for rapid modeling by parts. A generic model is used for modeling fitting according to the input image. User interface facilitates fitting by dragging on parts of a pose-aligned wireframe model. The algorithm estimates the "best" 3D movement of model parts based on their constraints. Changes to one part of the vehicle model are propagated to other parts through connectivity and symmetry constraints and a scattered data interpolation. The distribution of part changes to other parts prevents gaps in the model and speeds up the interaction process.

Lodha *et al.* proposed [39, 40] to train an AdaBoost classifier to perform building, grass and tree classification. They use height, height variation, normal variation and return intensity from LiDAR data as the features. A region-growing segmentation is used to obtain initial segmentation. Then, the region-wide information is again used to improve classification to deal with multiple returns and edge noise. A min-ε graph algorithm is

proposed to obtain initial footprint approximation. Finally, a Bayesian optimization is used to encode the prior information in terms of local angles, i.e., preference for 90 degrees (right angle) and 180 degrees (straight line).

Funded by DARPA under UrbanScape program, Marc Pollefeys and David Nister and their collaborators from UNC Chapel Hill and University of Kentucky proposed a system [41, 10] for automatic, geo-registered, real-time 3D reconstruction from video of urban scenes. The system collects video streams, as well as GPS and inertia measurements in order to place the reconstructed models in geo-registered coordinates. It is designed using current state of the art real-time modules for all processing steps. It employs commodity graphics hardware and standard CPU's to achieve real-time performance.

The Graphics and Media Lab in M. V. Lomonosov Moscow State University is also active on the development of image-based 3D reconstruction approach [42, 43, 44, 45, 46, 47, 48]. They focus on relatively fast, easy and inexpensive creation of photorealist urban models with medium detailed geometry and high quality textures, sufficient for usage in wide range of services directed toward common users. Their main intention was the creation of a system for construction high-quality models with very simple and intuitive user interface. Their reconstruction process includes the following steps: automatic tilt correction, semi-automatic segmentation of buildings, semi-automatic texture correction, and automatic 3D model construction. They have also developed an approach for semi-automatic registration of images for reconstruction of buildings from multiple views.

The group led by Ioannis Stamos from CUNY is fully dedicated to photorealistic 3D modeling. They [49, 50, 51, 52, 53, 54, 55] use 3D scanner to obtain range data. Then, registration of 3D range data and 2D images is performed. Finally, a mesh-simplification method of the final 3D model based on the segmentation results of each range image is developed for modeling.

There is a large literature on image-based building modeling. We review several studies without being exhaustive.

Single-view methods.

Criminisi *et al.* from Oxford [56] worked on single view reconstruction to reconstruct a scene from recognizable geometric primitives such as lines, planes and spheres by computing their spatial layout given only one view. Oh *et al.* from MIT [57] presented an interactive system to create models from a single image by manually assigning the depth based on a painting metaphor. Generally, these methods need intensive user interactions to produce visual pleasing results. Hoiem *et al.* from CMU [58] proposed surface lay-

out estimation for modeling. Saxena *et al.* from Stanford [59] and Torralba *et al.* from MIT [60, 61] learned the mapping information between image features and depth directly. Mueller *et al.* from ETH focus on image-based procedural modeling. Their approach relied on repetitive patterns on rectified texture image to discover façade structures, and obtained depth from manual input. Barinova *et al.* from Moscow State University [62] made used of manhattan structure for man-made building to divide the model fitting problem into chain graph inference. However, these approaches can only produce a rough shape for modeling objects without lots of details.

Interactive multi-view methods.

Façade, developed by Debevec *et al.* [15], is a seminal work in this category. They used line segment features in images and polyhedral blocks as 3D primitives to interactively register images and to reconstruct blocks with view-dependent texture mapping. However, the required manual selection of features and correspondences in different views is tedious, which makes it difficult to be scaled up when the number of images grows. van den Hengel *et al.* [63] used a sketching approach in one or more images to model a general object. But it is difficult to use this approach for detail modeling even with intensive interaction. Xiao *et al.* [64] proposed to approximate orthographic image by fronto-parallel reference image for each façade during automatic detail reconstruction and interactive user refinement. Therefore, their approach requires an accurate initialization and boundary for each façade as input, probably manually specified by the user. Sinha *et al.* [65] used registered multiple views and extracted the major directions by vanishing points. The significant user interactions required by these two methods for good results make them difficult to adopt in large-scale city modeling applications.

Automatic multi-view methods.

Werner and Zisserman [66] used line segments for building reconstruction from registered images by sparse points. Schindler *et al.* [67] proposed the use of line features for both structure from motion and modeling. Dick *et al.* [68] developed a 3D modeling architectural modeling method for short image sequences. The user is required to provide intensive architectural rules for the Bayesian inference. Many researchers realized the importance of line features in man-made scenes. However, line features tend to be sparse and geometrically less stable than points. Schindler and Bauer [69, 70, 71] is similar with that developed in [66]. A more systematic approach to modeling urban environments using video cameras has been investigated by several teams [41, 72]. They have been very successful in developing real-time video registration and focused on the global reconstruction

of dense stereo results from the registered images. Cornelis *et al.* [73, 74] also included a recognition component for cars and pedestrians to enhance the modeling process. Finally, the work by Schindler *et al.* [67] used the line-based structure from motion for urban environments. Zebedin *et al.* [75] is a representative of city modeling from aerial images, which is complementary to our reconstruction approach from street-level images.

Related projects

The literature related to image-based building modeling is very large. We only highlight some projects related to our approach.

Photo browsing. Snavely *et al.* [76, 77, 78, 79] proposed a system for browsing large collections of photographs in 3D. They used structure from motion technique as first step for camera calibration. This research project has been implemented as a commercial product – Microsoft Photosynth.

Street-view reconstruction. Mičušik and Košecká [80] has proposed to do 3D reconstruct using spherical images. The same date of Pittsburgh from Google Maps Street-view is used for evaluation of the method in this thesis.

Manhattan structure. Man-made structures always display Manhattan properties in structure. Among many researches utilizing this approach, Furukawa *et al.* [81, 82] and Gallup *et al.* [83] are especially inspiring.

Image parsing of buildings. Han and Zhu [84] proposed an image parsing of man-made objects in terms of rectangle primitives, which can be used to reconstruct buildings from a single image. Berg *et al.* [85] dedicated on parsing images of architectural scenes. Košecká *et al.* [86, 87, 88, 89] also dedicates on building parsing. The on-going eTRIMS (E-Training for Interpreting Images of Man-Made Scenes) project led by W. Förstner from University of Bonn, B. Neumann from University of Hamburg R. Šára from Czech Technical University in Prague, M. Petrou from Imperial College London, L. Hotz from HITEC Hamburg, concentrates on structural learning, where relations between components and compositional hierarchies play a central role in object categorization. Such learning is particularly relevant for the interpretation of man-made objects. Their project will use the recognition of buildings in outdoor scenes as its exemplary application domain.

1.2.2 Commercial products

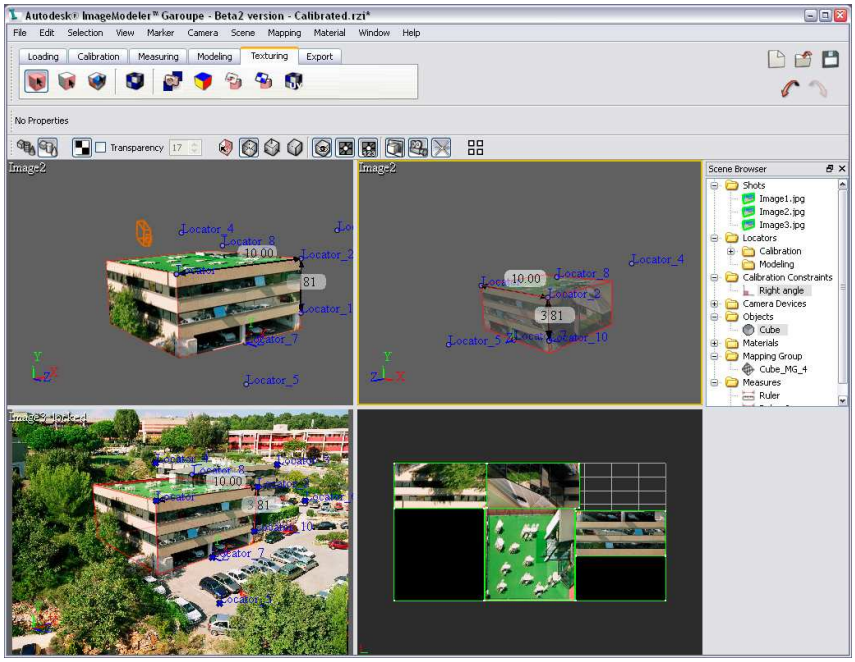
Autodesk ImageModeler and Eos Systems PhotoModeler are two software products that allow creation of texture-mapped 3-D models from a small number of photographs. Both of them requires the user to use mouse to click one point in one photograph, and click another point in another photograph, in order to provide point-to-point correspondences across different views. Given enough number of these manual correspondences, the software can calibrate the camera models for each photograph, and compute the 3D coordinate of these key points. After that, the user can draw mesh by linking key points.

UZR GmbH & Co KG iModeller 3D, D Vision Works D Sculptor, and Creative Dimension Software 3D Software Object Modeller are three software products that use a predefined and fixed patterns of geometry elements placed on the scene to enable camera calibration. They require a mask in some photographs to indicate the outline silhouette of the object of interest. There masks are used to compute a intersection in 3D by visual hull algorithm to produce a 3D model. The mesh is produced based on the polygon of intersections.

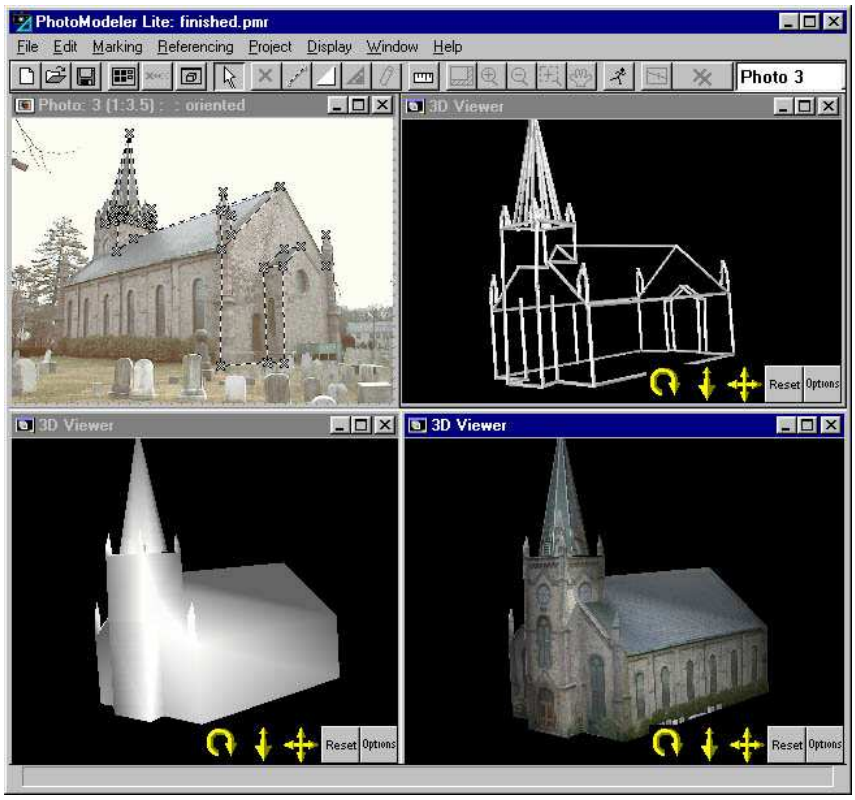
There are several industrial software systems for tracking objects in video or film and solving for 3D motion to allow for precise augmentation with 3D computer graphics, such as Vicon boujou, The Pixel Farm PFTrack, Andersson Technologies LLC SynthEyes, Science.D.Visions 3D Equalizer, Scenespector Systems VooCAT, Autodesk Maya Live. Vicon boujou also has modeling ability by triangulation of selected 3D points. The Pixel Farm PFTrack provides more image-based modeling functions similar with Autodesk ImageModeler. Several companies, such as 2d3, Imagineer Systems, MirriAd, Mova, Orad, Ooyala, PVI and the like, also use related tracking techniques for interesting applications from film and video, including new object composition, augmentation with 3D computer graphics, virtual advertising insertion, virtual television sets creation, visually assisted landing system, sports analysis, super resolution, image mosaicing, and stabilization etc.

1.3 Overview and organization

With enough human effort, most of the state-of-the-art approaches can achieve satisfactory results for most applications. Therefore, the remaining open question is: how to achieve the similar quality with less human effort, i.e., how to improve working efficiency. This thesis present an automatic approach that only requires minimal human efforts for image-based building modeling to achieve visual pleasing results. Our image-based approach has three steps: reconstruction, segmentation and modeling. The first step is to reconstruct a

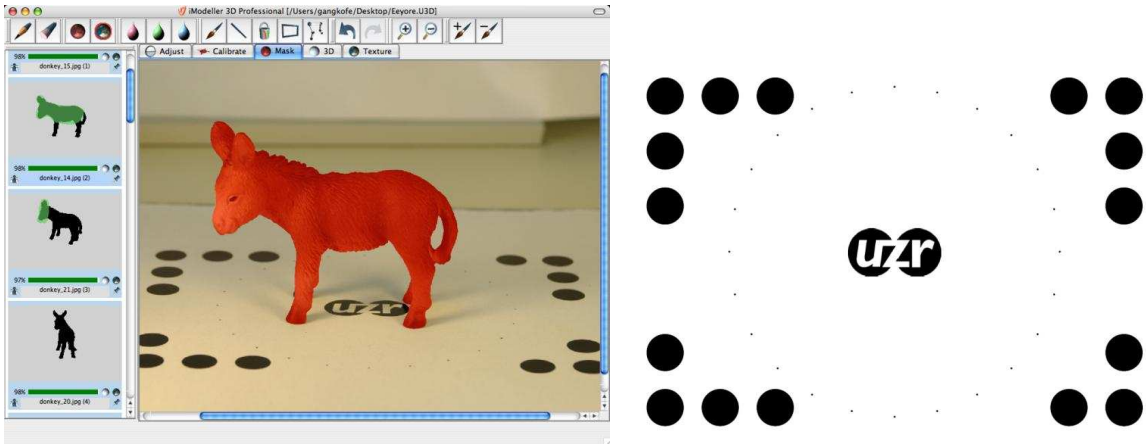


(a) User interface of Autodesk ImageModeler

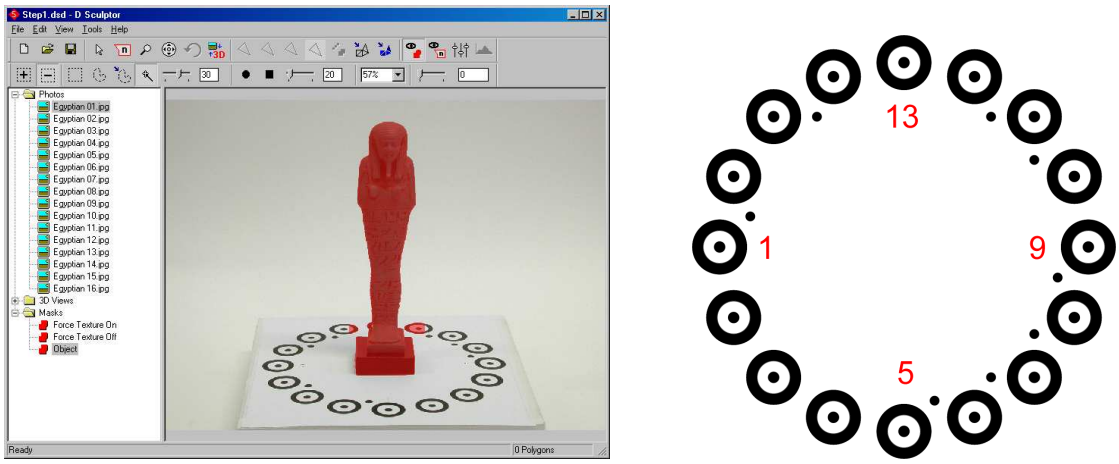


(b) User interface of Eos Systems PhotoModeler

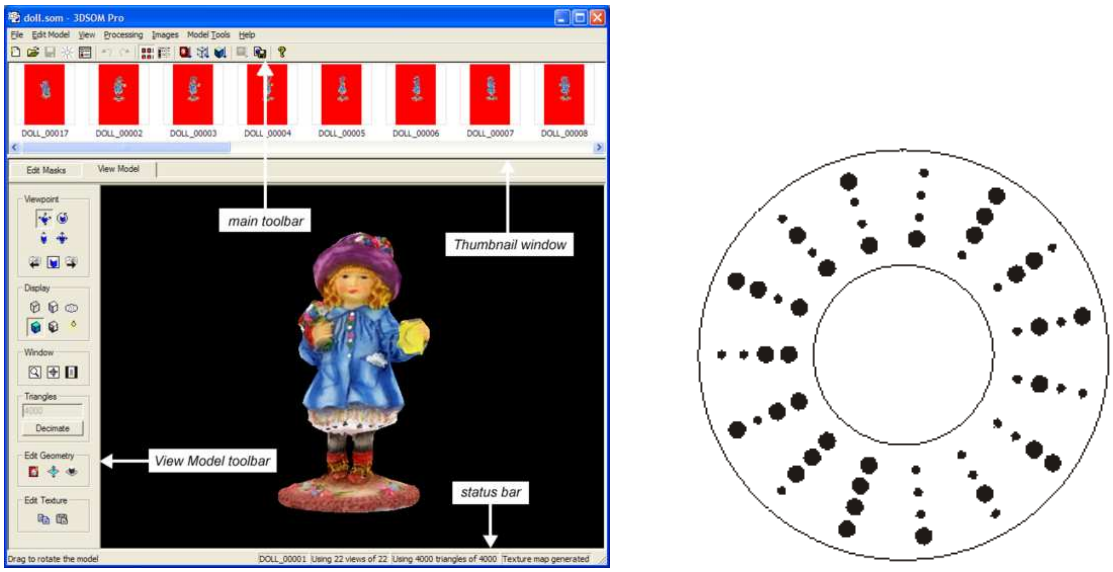
Figure 1.2: **User interface for manual correspondences.**



(a) UZR GmbH & Co KG iModeller 3D

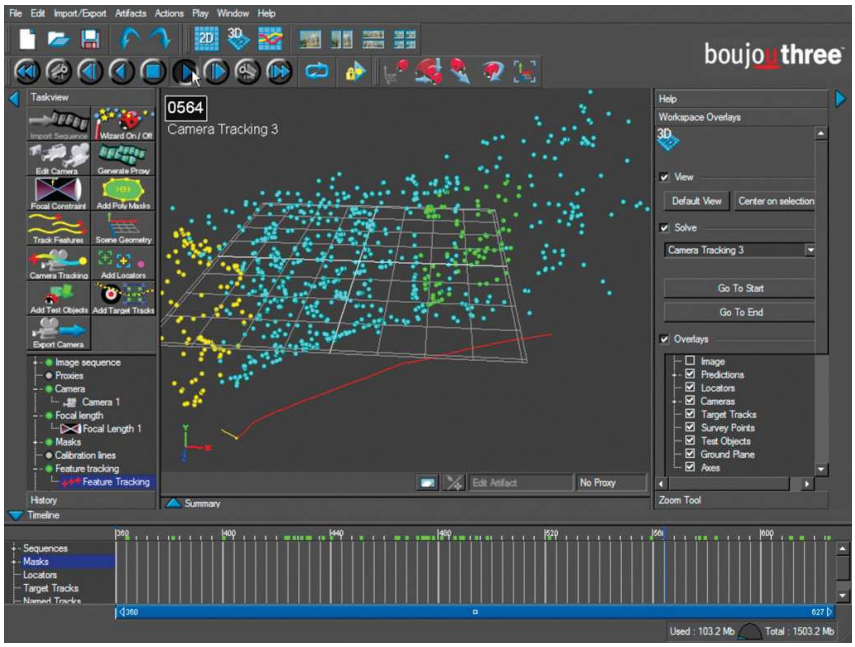


(b) D Vision Works D Sculptor

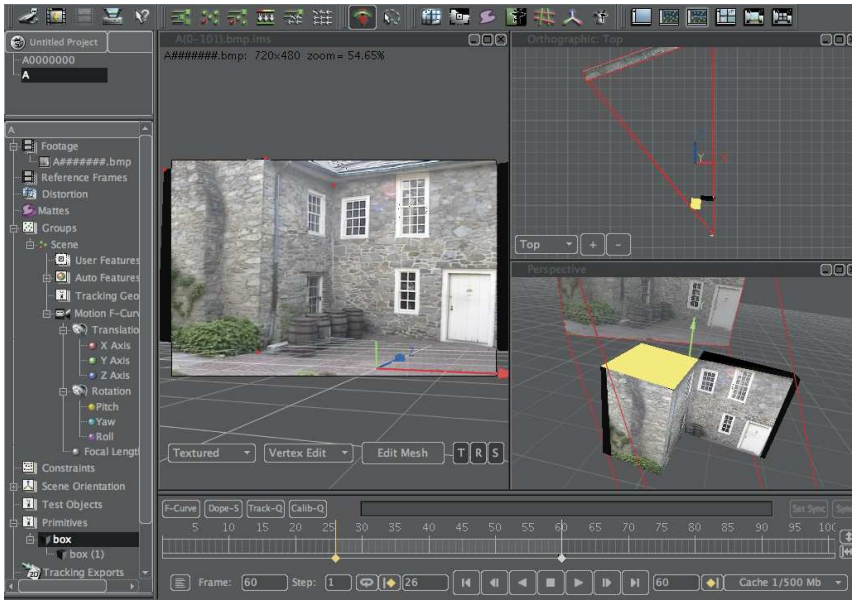


(c) Creative Dimension Software 3D Software Object Modeller

Figure 1.3: **User interface (left) and marker pattern (right).**



(a) Vicon boujou



(b) The Pixel Farm PFTrack

Figure 1.4: **User interface of popular matchmoving software.**

3D point cloud from the 2D images. The second step is to segment the 3D point cloud and the 2D images, where each group represents an object or a kind of objects for modeling. The final step is to model each object by structure analysis and regularization. This three-step division is very important, because it clearly divides the work into subproblems properly, and conquers each subproblem with strategies according to different objectives to be achieved in each stage. Therefore, in this thesis, we have three chapters, Chapter 2, 3, and 4, to describe the detail of each step. Finally, we evaluate our approach in Chapter 5, and conclude in Chapter 6.

CHAPTER 2

RECONSTRUCTION

The purpose of reconstruction is to reconstruct a 3D point cloud from the 2D images and obtain all the parameters of the cameras corresponding to each image. To achieve this objective, we first obtain pixel-to-pixel correspondences across two or three images. Then, we can reconstruct the scene and calibrate the cameras using the algorithms from multiple view geometry [90]. The multiple view geometry is a well-studied research in the past 20 years of computer vision community, and has achieved significant results to enable us to robustly reconstruct the scene. Since this reconstruction is only available to two or three images, we need to merge the reconstruction into a complete scene that contains all 3D points and camera positions in the same coordinate. Therefore, a sequence ordering strategy is needed to merge short sequences into long sequence. In this chapter, we first focus on point-to-point matching across two images. The matching algorithm is greatly affected on whether camera parameters are known before hand, for example, given by GPS/INS sensors. In Section 2.1, we introduce a robust point-to-point matching approach for uncalibrated cameras when no camera information is available. In Section 2.2, when the cameras are known, we address the computation speed problem using parallel computing technique on Graphics Processing Unit. Then, the ordering strategy for sequence merging is discussed for building modeling in Section 2.3.

2.1 Uncalibrated dense matching

2.1.1 Introduction

Stereo matching between images is a fundamental problem for 3D reconstruction. Here, we focus on matching two wide-baseline images taken from the same static scene. Unlike many previous methods which require that the input images be either calibrated [91] or rectified [92], we consider here a more challenging scenario in which the input contains two images only without any camera information. As a consequence, our method can be used for more general applications, such as motion estimation from structure.

Related work

Many stereo matching algorithms have been developed. Traditional stereo matching algorithms [92] were primarily designed for view pairs with a small baseline, and cannot be extended easily when the epipolar lines are not parallel. On the other hand, existing wide-baseline methods [93] depend heavily on the epipolar geometry which has to be provided, often through off-line calibration, while other methods can only recover very sparse matching [94, 95].

Although the epipolar geometry could be estimated on-line, those approaches still fail frequently for wide-baseline image pairs since the sparse matching result is fragile and the estimated fundamental matrix often fits only to some parts of the image but not the entire image. Region growing based methods [96, 97] can achieve denser matching, but may easily get trapped in local optima. Therefore its matching quality depends heavily on the result of the initial sparse matching. Also, for image pairs with quite different pixel scales, it is very difficult to achieve reasonable results due to discrete growing.

Recent research shows that learning techniques can improve the performance of matching by taking matched pairs as training data or by learning the probabilistic image prior [98] that encodes the smoothness constraint for natural images. However, for a test image pair, the information learned from other irrelevant images is very weak in the sense that it is unrelated to the test image pair. Thus the quality of the result greatly depends on the training data.

Our approach

In this work, we explore the dense matching of uncalibrated wide-baseline images by utilizing all the local, regional and global information simultaneously in an optimization procedure. We propose a semi-supervised approach to the matching problem requiring only two input images taken from the same static scene. Since the method does not rely on any training data, it can handle images from any scene with stable performance.

We consider two data sets, $\mathcal{X}^1$ and $\mathcal{X}^2$, corresponding to the two input images with $n^1 = r^1 \times c^1$ pixels and $n^2 = r^2 \times c^2$ pixels, respectively. For $p = 1, 2$,

$$\mathbf{X}^p = \left(\mathrm{x}_1^p, \mathrm{x}_2^p, \ldots, \mathrm{x}_{(s^p-1)\times c^p + t^p}^p, \ldots, \mathrm{x}_{n^p}^p\right)^T, \tag{2.1}$$

where $\mathrm{x}_{(s^p-1)\times c^p + t^p}^p$ represents the pixel located at the coordinate position (s^p, t^p) in the p-th image space, $s^p \in \{1, \cdots, r^p\}$, and $t^p \in \{1, \cdots, c^p\}$. Here, we define $q = 3 - p$, meaning that $q = 1$ when $p = 2$ and $q = 2$ when $p = 1$, and let $i = (s^p - 1) \times c^p + t^p$.

For each pixel x_i^p, we want to find a matching point located at coordinate position (s^q, t^q) in the q-th (continuous) image space, where $s^q, t^q \in \mathcal{R}$. Hence, we can use a label vector to represent the position offset from a point in the second image to the corresponding point in the first image: $\mathrm{y}_i^p = (v_i^p, h_i^p)^T = ((s^1, t^1) - (s^2, t^2))^T \in \mathcal{R}^2$. In this way, our label vector representation takes real numbers for both elements, thus supporting sub-pixel matching. Let $\mathbf{Y}^p = (\mathrm{y}_1^p, \cdots, \mathrm{y}_{n^p}^p)^T$ be the label matrix, and $\mathrm{O}^p = (o_1^p, \cdots, o_{n^p}^p)^T$ be the corresponding visibility vector: $o_i^p \in [0, 1]$ is close to 1 if the 3D point corresponding to the data point x_i^p is visible in the other image, and otherwise close to 0 such as a point in the occluded region. This notion of visibility may also be interpreted as matching confidence.

Obviously, nearby pixels are more likely to have similar label vectors. This smoothness constraint, relying on the position of the data points, can be naturally represented by a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ where the node set $\mathcal{V}$ represents the data points and the edge set $\mathcal{E}$ represents the affinities between them. In our setting, we have two graphs $\mathcal{G}^1 = \langle \mathcal{V}^1, \mathcal{E}^1 \rangle$ and $\mathcal{G}^2 = \langle \mathcal{V}^2, \mathcal{E}^2 \rangle$ for the two images where $\mathcal{V}^1 = \{\mathrm{x}_i^1\}$ and $\mathcal{V}^2 = \{\mathrm{x}_i^2\}$. Let $\mathcal{N}(\mathrm{x}_i^p)$ be the set of data points in the neighborhood of x_i^p. The affinities can be represented by two weight matrices $\mathbf{W}^1$ and $\mathbf{W}^2$: w_{ij}^p is non-zero iff x_i^p and x_j^p are neighbors in $\mathcal{E}^p$.

In recent years, matching techniques such as SIFT [94] are powerful enough to recover some sparsely matched pairs. Now, the problem here is, given such matched pairs as labeled data $\langle \mathbf{X}_l^1, \mathbf{Y}_l^1 \rangle$, $\langle \mathbf{X}_l^2, \mathbf{Y}_l^2 \rangle$ and the affinity matrices $\mathbf{W}^1$ and $\mathbf{W}^2$, we want to infer the label matrices for the remaining unlabeled data $\langle \mathbf{X}_u^1, \mathbf{Y}_u^1 \rangle, \langle \mathbf{X}_u^2, \mathbf{Y}_u^2 \rangle$. For the sake of clarity of presentation and without loss of generality, we assume that the indices of the data points are arranged in such a way that the labeled points come before the unlabeled ones, that is $\mathbf{X}^p = \left((\mathbf{X}_l^p)^T, (\mathbf{X}_u^p)^T\right)^T$. For computation, the index of the data point can be mapped by multiplying elementary matrices for row-switching transformations.

In what follows, we formulate in Section 2.1.2 the matching problem under a graph-based semi-supervised label propagation framework, and solve the optimization problem via an iterative cost minimization procedure in Section 2.1.3. To get reliable affinity matrices for propagation, in Section 2.1.4 we learn $\mathbf{W}^1$ and $\mathbf{W}^2$ directly from the input images which include color and depth information. The complete procedure of our algorithm is summarized in Algorithm 1. More details are given in Section 2.1.5. Finally, extensive experimental results are presented in Section 2.2.5.

2.1.2 Semi-supervised matching framework

Semi-supervised learning on the graph representation tries to find a label matrix $\hat{\mathbf{Y}}^p$ that is consistent with both the initial incomplete label matrix and the geometry of the data

manifold induced by the graph structure. Because the incomplete labels may be noisy, the estimated label matrix $\hat{\mathbf{Y}}_l^p$ for the labeled data is allowed to differ from the given label matrix $\mathbf{Y}_l^p$. Given an estimated $\hat{\mathbf{Y}}^p$, consistency with the initial labeling can be measured by

$$C_l^p\left(\hat{\mathbf{Y}}^p, \mathrm{O}^p\right) = \sum_{\mathrm{x}_i^p \in \mathcal{X}_l^p} o_i^p \left\|\hat{\mathrm{y}}_i^p - \mathrm{y}_i^p\right\|^2. \tag{2.2}$$

On the other hand, consistency with the geometry of the data in the image space, which follows from the smooth manifold assumption, motivates a penalty term of the form

$$C_s^p\left(\hat{\mathbf{Y}}^p, \mathrm{O}^p\right) = \frac{1}{2} \sum_{\mathrm{x}_i^p, \mathrm{x}_j^p \in \mathcal{X}^p} w_{ij}^p \phi\left(o_i^p, o_j^p\right) \left\|\hat{\mathrm{y}}_i^p - \hat{\mathrm{y}}_j^p\right\|^2, \tag{2.3}$$

where $\phi\left(o_i^p, o_j^p\right) = \frac{1}{2}\left(\left(o_i^p\right)^2 + \left(o_j^p\right)^2\right)$. When o_i^p and o_j^p are both close to 1, the function value is also close to 1. This means we penalize rapid changes in $\hat{\mathbf{Y}}^p$ between points that are close to each other (as given by the similarity matrix $\mathbf{W}^p$), and only enforce smoothness within visible regions, i.e., o^p is large.

Local label preference cost

Intuitively, two points of a matched pair in the two images should have great similarity in terms of the features since they are two observations of the same 3D point. Here, we use a similarity cost function $\rho_i^p(\mathrm{y})$ to represent the similarity cost between the pixel x_i^p in one image and the corresponding point for the label vector y in the other image space (detailed in Subsection 2.1.5). On the other hand, if o_i^p is close to 0, which means that x_i^p is almost invisible and the matching has low confidence, the similarity cost should not be charged. To avoid the situation when every point tends to have zero visibility to prevent cost charging, we introduce a penalty term τ_i^p. When o_i^p is close to 0, $(1 - o_i^p)\,\tau_i^p$ will increase. Also, τ_i^p should be different for different x_i^p. Textureless regions should be allowed to have lower matching confidence, that is, small confidence penalty, and vice versa. We use a very simple difference-based confidence measure defined as follows

$$\tau_i^p = \max_{\mathrm{x}_j^p \in \mathcal{N}\left(\mathrm{x}_i^p\right)} \left\{\left\|\mathrm{x}_i^p - \mathrm{x}_j^p\right\|\right\}. \tag{2.4}$$

Now, we can define the local cost as

$$C_d^p\left(\hat{\mathbf{Y}}^p, \mathrm{O}^p\right) = \sum_{\mathrm{x}_i^p \in \mathcal{X}^p} \left(o_i^p \rho_i^p\left(\hat{\mathrm{y}}_i^p\right) + \left(1 - o_i^p\right)\tau_i^p\right). \tag{2.5}$$

Regional surface shape cost

The shapes of the 3D objects' surfaces in the scene are very important cues for matching. An intuitive approach is to use some methods based on two-view geometry to reconstruct the 3D surfaces. While this is a reasonable choice, it is unstable since the structure deduced from two-view geometry is not robust especially when the baseline is not large enough. Instead, we adopt the piecewise planar patch assumption [97]. Since two data points with high affinity relation are more likely to have similar label vectors, we assume that the label vector of a data point can be linearly approximated by the label vectors of its neighbors, as in the manifold learning method called locally linear embedding (LLE) [99], that is

$$\mathrm{y}_i^p = \sum_{\mathrm{x}_j^p \in \mathcal{N}\left(\mathrm{x}_i^p\right)} w_{ij}^p \mathrm{y}_j^p. \tag{2.6}$$

Hence, the reconstruction cost can be defined as

$$C_r\left(\mathbf{Y}^p\right) = \sum_{\mathrm{x}_i^p \in \mathcal{X}^p} \left\| \mathrm{y}_i^p - \sum_{x_j^p \in \mathcal{N}\left(\mathrm{x}_i^p\right)} w_{ij}^p \mathrm{y}_j^p \right\|^2 = \left\| \left(\mathbf{I} - \mathbf{W}^p\right) \mathbf{Y}^p \right\|_F^2. \tag{2.7}$$

Let $\mathbf{A}^p = \mathbf{W}^p + \left(\mathbf{W}^p\right)^T - \mathbf{W}^p \left(\mathbf{W}^p\right)^T$ be the adjacency matrix, $\mathbf{D}^p$ the diagonal matrix containing the row sums of the adjacency matrix $\mathbf{A}^p$, and $\mathbf{L}^p = \mathbf{D}^p - \mathbf{A}^p$ the un-normalized graph Laplacian matrix. Because of the way $\mathbf{W}^p$ is defined in Section 2.1.4, we have $\mathbf{D}^p \approx \mathbf{I}$. Therefore,

$$C_r\left(\mathbf{Y}^p\right) \approx \mathrm{tr}\left(\left(\mathbf{Y}^p\right)^T \mathbf{L}^p \mathbf{Y}^p\right) = \sum_{\mathrm{x}_i^p, \mathrm{x}_j^p \in \mathcal{X}^p} a_{ij}^p \left\| \mathrm{y}_i^p - \mathrm{y}_j^p \right\|^2. \tag{2.8}$$

This approximation induces the representation of $a_{ij}^p \left\| \mathrm{y}_i^p - \mathrm{y}_j^p \right\|^2$, which makes the integration of the cost with visibility much easier.

Now, the data points from each image lie on one 2D manifold (image space). Except for the occluded parts which cannot be matched, the two 2D manifolds are from the same 2D manifold of the visible surface of the 3D scene. LLE [100] is used to align the two 2D manifolds (image spaces) to one 2D manifold (visible surface). The labeled data (known matched pairs) are accounted for by constraining the mapped coordinates of matched points to coincide. Let $\mathcal{X}_c^p = \mathcal{X}_l^p \cup \mathcal{X}_u^p \cup \mathcal{X}_u^q$, $\hat{\mathbf{Y}}_c^p = \left(\left(\hat{\mathbf{Y}}_l^p\right)^T, \left(\hat{\mathbf{Y}}_u^p\right)^T, \left(\hat{\mathbf{Y}}_u^q\right)^T\right)^T$ and $\mathrm{O}_c^p = \left(\left(\mathrm{O}_l^p\right)^T, \left(\mathrm{O}_u^p\right)^T, \left(\mathrm{O}_u^q\right)^T\right)^T$. We partition $\mathbf{A}^p$ as

$$\mathbf{A}^p = \begin{bmatrix} \mathbf{A}_{ll}^p & \mathbf{A}_{lu}^p \\ \mathbf{A}_{ul}^p & \mathbf{A}_{uu}^p \end{bmatrix}. \tag{2.9}$$

Alignment of the manifold can be done by combining the Laplacian matrix as in [100], which is equivalent to combining the adjacency matrix:

$$\mathbf{A}_c^p = \begin{bmatrix} \mathbf{A}_{ll}^p + \mathbf{A}_{ll}^q & \mathbf{A}_{lu}^p & \mathbf{A}_{lu}^q \\ \mathbf{A}_{ul}^p & \mathbf{A}_{uu}^p & 0 \\ \mathbf{A}_{ul}^q & 0 & \mathbf{A}_{uu}^q \end{bmatrix}. \tag{2.10}$$

Imposing the cost only on the visible data points, the separate LLE cost of each graph is summed up:

$$C_r^p\left(\hat{\mathbf{Y}}^1, \hat{\mathbf{Y}}^2, \mathrm{O}^1, \mathrm{O}^2\right) = \sum_{\mathrm{x}_i^p, \mathrm{x}_j^p \in \mathcal{X}_c^p} (a_c^p)_{ij}\, \phi\big((o_c^p)_i, (o_c^p)_j\big) \left\| (\hat{\mathrm{y}}_c^p)_i - (\hat{\mathrm{y}}_c^p)_j \right\|^2, \tag{2.11}$$

where $(a_c^p)_{ij}$ is the element of $\mathbf{A}_c^p$.

Global epipolar geometric cost

In the epipolar geometry [90], the fundamental matrix $\mathbf{F}_{12} = \mathbf{F}_{21}^T$ encapsulates the intrinsic projective geometry between two views in the way that, for x_i^p at position (s^p, t^p) in one image with matching point at position (s^q, t^q) in the other image, the matching point (s^q, t^q) should lie on the line $(a_i^p, b_i^p, c_i^p) = (s^p, t^p, 1)\,\mathbf{F}_{pq}^T$. This global constraint affects every matching pair in the two images. For x_i^p, we define $d_i^p(\mathrm{y})$ to be the squared Euclidean distance in the image space of the other image between the corresponding epipolar line (a_i^p, b_i^p, c_i^p) and the matching point (s^q, t^q):

$$d_i^p(\mathrm{y}) = \frac{\left(a_i^p s^q + b_i^p t^q + c_i^p\right)^2}{\left(a_i^p\right)^2 + \left(b_i^p\right)^2}, \tag{2.12}$$

where $\mathrm{y} = (v, h)^T = \left((s^1 - s^2), (t^1 - t^2)\right)^T$. The global cost is now the sum of all squared distances:

$$C_g^p\left(\hat{\mathbf{Y}}^p, O^p\right) = \sum_{\mathrm{x}_i^p \in \mathcal{X}^p} o_i^p d_i^p(\hat{\mathrm{y}}_i^p). \tag{2.13}$$

Symmetric visibility consistency cost

Assume that x_i^p in one image is matched with x_j^q in the other image. x_j^q should also have a label vector showing its matching with x_i^p in the original image. This symmetric visibility

consistency constraint motivates the following visibility cost

$$C_v^p\left(\mathrm{O}^p, \hat{\mathrm{Y}}^q\right) = \beta \sum_{\mathrm{x}_i^p \in \mathcal{X}^p} \left(o_i^p - \gamma_i^p\left(\hat{\mathrm{Y}}^q\right)\right)^2 + \frac{1}{2} \sum_{\mathrm{x}_i^p, \mathrm{x}_j^p \in \mathcal{X}^p} w_{ij}^p \left(o_i^p - o_j^p\right)^2, \tag{2.14}$$

where $\gamma\left(\hat{\mathrm{Y}}^q\right)$ is a function defined on the p-th image space. For each x_i^p, its value via the γ function indicates whether or not there exist one or more data points that match a point near x_i^p from the other view according to $\hat{\mathrm{Y}}^q$. The value at pixel x_i^p is close to 0 if there is no point in the other view corresponding to a point near x_i^p, and otherwise close to 1. The parameter β controls the strength of the visibility constraint. The last term enforces the smoothness of the occlusion that encourages spatial coherence and is helpful to remove some isolated pixels or small holes of the occlusion.

The γ function can be computed as a voting procedure when $\hat{\mathrm{Y}}^q$ is available in the other view. For each point x_j^q at position (s^q, t^q) in $\mathcal{X}^q$ with label $\mathrm{y}_j^q = \left(v_j^q, h_j^q\right)^T = \left((s^1, t^1) - (s^2, t^2)\right)^T$, equivalent to be matched with a point at position (s^p, t^p), we place a 2D Gaussian function $\psi\left(s, t\right)$ on the p-th image centered at the matched position $\mathrm{c}_j = \left(s^p, t^p\right)^T$. Now, we get a Gaussian mixture model $\sum_{\mathrm{x}_j^q} \psi_{\mathrm{c}_j}\left(s, t\right)$ in the voted image space. Truncating it, we get

$$\gamma^p\left(s, t\right) = \min\left\{1, \sum_{\mathrm{x}_j^q \in \mathcal{X}^q} \psi_{\mathrm{c}_j}\left(s, t\right)\right\}. \tag{2.15}$$

Our matching framework combines all the costs described above. We now present our iterative optimization algorithm to minimize the costs.

2.1.3 Iterative MV optimization

It is intractable to minimize the matching and visibility costs simultaneously. Therefore, our optimization procedure iterates between two steps: 1) the M-step estimates matching given visibility, and 2) the V-step estimates visibility given matching. Before each iteration, we estimate the fundamental matrix $\mathbf{F}$ by the normalized 8-point algorithm with RANSAC followed by the gold standard algorithm that uses the Levenberg-Marquardt algorithm to minimize the geometric distance [90]. Then, we use $\mathbf{F}$ to reject the outliers from the matching result of the previous iteration and obtain a set of inliers as the initial labeled data points. The iterations stop when the cost difference between two consecutive iterations is smaller than a threshold, which means that the current matching result is already quite stable. The whole iterative optimization procedure is summarized in Algorithm 1.

Algorithm 1 The complete procedure.

1. Compute the depth and occlusion boundary images and feature vectors (Section 2.1.5).
2. Compute sparse matching by SIFT and the confidence penalty τ, then interpolate the results from sparse matching with depth information to obtain an initial solution (Subsection 2.1.5).
3. Learn the affinity matrices $\mathbf{W}^1$ and $\mathbf{W}^2$ (Section 2.1.4).
4. while (cost change between two iterations $\geq$ threshold):
   (a) Estimate the fundamental matrix $\mathbf{F}$, and reject outliers to get a subset as labeled data (Section 2.1.3),
   (b) Compute the parameters for the similarity cost function ρ and epipolar cost function d (Subsection 2.1.5 and 2.1.2),
   (c) Estimate matching given visibility (Subsection 2.1.3),
   (d) Compute the γ map (Subsection 2.1.2),
   (e) Estimate visibility given matching (Subsection 2.1.3).

M-step: estimation of matching given visibility

Actually, the visibility term C_v imposes two kinds of constraints on the matching $\hat{\mathbf{Y}}$ given the visibility O: First, for each pixel x_i^p in the p-th image, it should not match the invisible (occluded) points in the other image. Second, for each visible pixel in the q-th image, at least one pixel in the p-th image should match its nearby points. The first restriction is a local constraint that is easy to satisfy. However, the second constraint is a global one on the matching of all points, which is implicitly enforced in the matching process. Therefore, in this step, we approximate the visibility term by considering only the local constraint [101], which means that some possible values for a label vector, corresponding to the occluded region, have higher costs than the other possible values. This variation of the cost can be incorporated into the similarity function $\rho_i^p(\mathrm{y})$ in C_d. Let $\mathbf{Y} = \left(\left(\mathbf{Y}^1\right)^T, \left(\mathbf{Y}^2\right)^T\right)^T$. Summing up all the costs and considering the two images together, our cost function is

$$C_M\left(\hat{\mathbf{Y}}\right) = \sum_{p=1,2} \left(\lambda_l C_l^p + \lambda_s C_s^p + \lambda_d C_d^p + \lambda_r C_r^p + \lambda_g C_g^p\right) + \epsilon\left\|\hat{\mathbf{Y}}\right\|^2, \tag{2.16}$$

where $\epsilon\left\|\hat{\mathbf{Y}}\right\|^2$ is a small regularization term to avoid reaching degenerate situations. Fixing O^1 and O^2, cost minimization is done by setting the derivative with respect to $\hat{\mathbf{Y}}$ to zero since the second derivative is a positive definite matrix.

V-step: estimation of visibility given matching

After achieving a matching, we can recompute the γ map (Subsection 2.1.2). Let $\mathrm{O} = \left(\left(\mathrm{O}^1\right)^T, \left(\mathrm{O}^2\right)^T\right)^T$. Then, summing up all the costs and considering the two images together, our cost function is

$$C_V\left(\mathrm{O}\right) = \sum_{p=1,2} \left(\lambda_l C_l^p + \lambda_s C_s^p + \lambda_d C_d^p + \lambda_r C_r^p + \lambda_g C_g^p + \lambda_v C_v^p\right) + \epsilon \left\|\mathrm{O}\right\|^2, \tag{2.17}$$

where $\epsilon \left\|\mathrm{O}\right\|^2$ is a small regularization term. Now, for fixed $\hat{\mathbf{Y}}^1$ and $\hat{\mathbf{Y}}^2$, cost minimization is done by setting the derivative with respect to O to zero since the second derivative is a positive definite matrix.

Since $\mathbf{W}^p$ is very sparse, the coefficient matrix of the system of linear equations is also very sparse in the above two steps. We use a Gauss-Seidel solver or a conjugate gradient method on GPU [102], which can solve in parallel a large sparse system of linear equations very efficiently. We can derive that by the way $\mathbf{W}^p$ is defined in Section 2.1.4 and the cost functions defined in Equation 2.16 and Equation 2.17, the coefficient matrix is strictly diagonally dominant and positive definite. Hence, both Gauss-Seidel and conjugate gradient converge to the solution of the linear system with theoretical guarantee.

2.1.4 Learning the symmetric affinity matrix

We have presented our framework which finds a solution by solving an optimization problem. Traditionally, for $\mathbf{W}^1$ and $\mathbf{W}^2$, we can directly define the pairwise affinity between two data points by normalizing their distance. However, as pointed out by [103], there exists no reliable approach for model selection if only very few labeled points are available, since it is very difficult to determine the optimal normalization parameters. Thus we prefer using a more reliable and stable way to learn the affinity matrices.

Similar to the 3D visible surface manifold of Equation 2.6 in Section 2.1.2, we make the smooth manifold and linear reconstruction assumptions for the manifold in the image space. We also assume that the label space and image space share the same local linear reconstruction weights. Then we can obtain the linear reconstruction weight matrix $\mathbf{W}^p$ by minimizing the energy function $E_{\mathbf{W}^p} = \sum_{\mathrm{x}_i^p \in \mathcal{X}^p} E_{\mathrm{x}_i^p}$, where

$$E_{\mathrm{x}_i^p} = \left\| \mathrm{x}_i^p - \sum_{\mathrm{x}_j^p \in \mathcal{N}\left(\mathrm{x}_i^p\right)} w_{ij}^p \mathrm{x}_j^p \right\|^2. \tag{2.18}$$

This objective function is similar to the one used in LLE [99], in which the low-dimensional coordinates are assumed to share the same linear reconstruction weights

with the high-dimensional coordinates. The difference here is that we assume the sharing relation to be between the label vectors and the features [104]. Hence, the way we construct the whole graph is to first shear the whole graph into a series of overlapped linear patches and then paste them together. To avoid the undesirable contribution of negative weights, we further enforce the following constraint

$$\sum_{\mathrm{x}_j^p \in \mathcal{N}\left(\mathrm{x}_i^p\right)} w_{ij}^p = 1,\ w_{ij}^p \geq 0. \tag{2.19}$$

From Equation 2.18, $E_{\mathrm{x}_i^p} = \sum_{\mathrm{x}_j^p, \mathrm{x}_k^p \in \mathcal{N}\left(\mathrm{x}_i^p\right)} w_{ij}^p \mathrm{G}_{jk}^i w_{ik}^p$, where $\mathrm{G}_{jk}^i = \left(\mathrm{x}_i^p - \mathrm{x}_j^p\right)^T \left(\mathrm{x}_i^p - \mathrm{x}_k^p\right)$. Obviously, the more similar is x_i^p to x_j^p, the larger will w_{ij}^p be. Also, w_{ij}^p and w_{ji}^p should be the same since they both correspond to the affinity relation between x_i^p and x_j^p. However, the above constraints do not either enforce or optimize to have this characteristic and the hard constraint $w_{ij}^p = w_{ji}^p$ may result in violation of Equation 2.19. Hence, we add a soft penalty term $\sum_{ij} \left(w_{ij}^p - w_{ji}^p\right)^2$ to the objective function. Thus the reconstruction weights of each data point can be obtained by solving the following quadratic programming (QP) problem

$$\begin{aligned} \min_{\mathrm{W}^p} & \sum_{\mathrm{x}_i^p \in \mathcal{X}^p} \sum_{\mathrm{x}_j^p, \mathrm{x}_k^p \in \mathcal{N}\left(\mathrm{x}_i^p\right)} w_{ij}^p \mathrm{G}_{jk}^i w_{ik}^p + \kappa \sum_{ij} \left(w_{ij}^p - w_{ji}^p\right)^2 \\ & \text{s.t. } \forall \mathrm{x}_i^p \in \mathcal{X}^p, \sum_{\mathrm{x}_j^p \in \mathcal{N}\left(\mathrm{x}_i^p\right)} w_{ij}^p = 1,\ w_{ij}^p \geq 0. \end{aligned} \tag{2.20}$$

After all the reconstruction weights are computed, two sparse matrices can be constructed by $\mathbf{W}^p = \left[w_{ij}^p\right]$ while letting $w_{ii}^p = 0$ for all x_i^p. In our experiment, $\mathbf{W}^p$ is almost symmetric and we further update it by $\mathbf{W}^p \leftarrow \frac{1}{2}\left((\mathbf{W}^p)^T + \mathbf{W}^p\right)$. Since the soft constraint has made $\mathbf{W}^p$ similar to $(\mathbf{W}^p)^T$, this update just changes $\mathbf{W}^p$ slightly, and will not lead to unreasonable artifacts. To achieve speedup, we can first partition the graph into several connected components by the depth information and super-pixel over-segmentation on the RGB image, and break down the large QP problem into several smaller QP problems with one QP for each connected component, then solve them one by one.

2.1.5 Initialization and similarity

The feature vectors are defined as RGB color. For each image, we recover the occlusion boundaries and depth ordering in the scene. The method in [105] is used to learn to identify and label occlusion boundaries using the traditional edge and region cues together

with 3D surface and depth cues. Then, from just a single image, we obtain a depth estimation and the occlusion boundaries of free-standing structures in the scene. We append the depth value to the feature vector.

Label initialization by depth

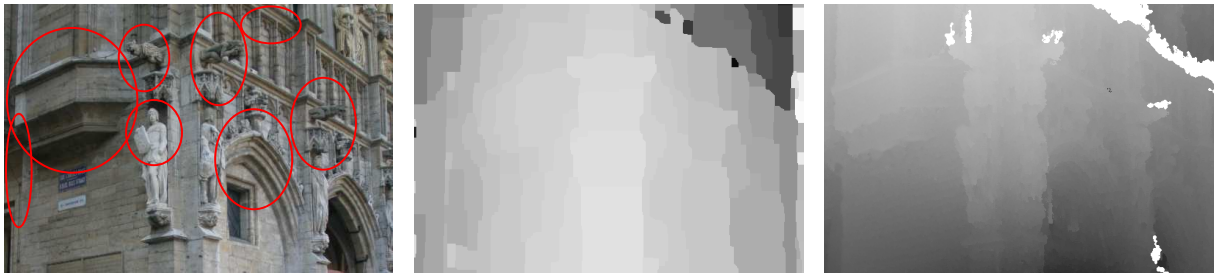
We use SIFT [94] and a nearest neighbor classifier to obtain an initial matching. For robustness, we perform one-to-one cross consistency checking, which matches points of the first image to the second image, and inversely matches points of the second image to the first image. Only the best matched pairs consistent in both directions are retained. To avoid errors on the occlusion boundary due to the similar color of background and foreground, we filter the sparse matching results and reject all pairs that are too close to the occlusion boundaries. Taking the remaining as seed points, with the depth information, region growing is used to achieve an initial dense matching [97]. Then, the remaining unmatched part is interpolated. Assuming the nearby pixels in the same partition lie on a planar surface, we estimate the homography transformation between two corresponding regions in the two images. With the estimated homography, the unknown regions are labeled and the occlusion regions are also estimated.

Computing the similarity cost function

As mentioned in Section 2.1.2, the continuous-valued similarity cost function $\rho_i^p(\mathrm{y})$ represents the difference between point x_i^p and the matching point, characterizing how suitable it is for x_i^p to have label $\mathrm{y} = (v, h)^T$. Since our algorithm works with some labeled data in a semi-supervised manner by the consistent cost C_l, the local cost C_d just plays a secondary role. Hence, unlike the traditional unsupervised matching [101], our framework does not heavily rely on the similarity function $\rho_i^p(\mathrm{y})$. Therefore, for efficient computation, we just sample some values for some integer combination of h and v to compute $\rho_i^p(\mathrm{y}) = \exp(-\frac{\left\|\mathrm{x}_i^p - \mathrm{x}_j^q\right\|^2}{2\sigma^2})$. We normalize the largest sampled value to 1, and then fit $\rho_i^p(\mathrm{y})$ with a continuous and differentiable quadratic function $\rho_i^p(\mathrm{y}) = \frac{(v-v_o)^2+(h-h_o)^2}{2\sigma^2}$, where (v_o, h_o) and σ are the center and spread of the parabola for x_i^p.

2.1.6 Experiments

In all our experiments performed on a desktop PC with Intel Core 2 Duo E6400 CPU and NVIDIA GeForce 8800 GTX GPU, the number of iterations is always less than 9 before stopping and the computation time is less than 41 seconds for each image pair,



(a) Input image [93] (b) Result from [1] (c) Result of our method

Figure 2.1: **Comparison with [1]**. Attention should be paid to the fine details outlined by red circles. Also, our method can correctly detect the occluded region and does not lead to block artifacts that are typically given by Graph-cut. Subfigure (b) is extracted from Figure 7 of [1]. The input dataset is from C. Strecha [2].

excluding the time spent on estimating the depth for a single image by [105]. We set the parameters to favor C_l and C_g in the M-step and C_v in the V-step. Since there is no ground truth in searching for good parameter values, we tune the parameters manually and then fix them for all experiments. To solve the QP problem for $\mathbf{W}^p$, we first compute a "warm start" without involving the positive constraints using the method in [99], and then run the active set algorithm on this "warm start", which converges rapidly in just a few iterations. We demonstrate our algorithm on various data sets in Figure 2.1.6, most of which have very complex shape with similar color that makes the matching problem very challenging. Compared with [1], our method can produce more detail, as shown in Figure 2.1.6. In the figures of the matching results, the intensity value is set to be the norm of the label vector, that is $\|\mathrm{y}\|$, and only visible matching with $o \geqslant 0.5$ is shown.

Application to 3-view reconstruction

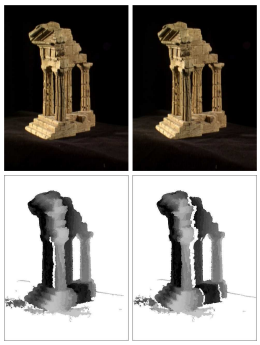
In our target application, we have no information about the camera. To produce a 3D reconstruction result, we use three images to recover the motion information. Five examples are shown in Figure 2.1.6. The proposed method is used to compute the point correspondence between the first and second images, as well as the second and third images. Taking the second image as a bridge, we can obtain the feature tracks of three views. As in [106], these feature tracks across three views are used to obtain projective reconstruction by [107], and are metric upgraded inside a RANSAC framework, followed by bundle adjustment [90]. Note that the feature tracks with too large reprojection errors are considered as outliers and are not shown in the 3D reconstruction result in Figure 2.1.6.

(a) City Hall Brussels

(b) Bookshelf

(c) Valbonne Church

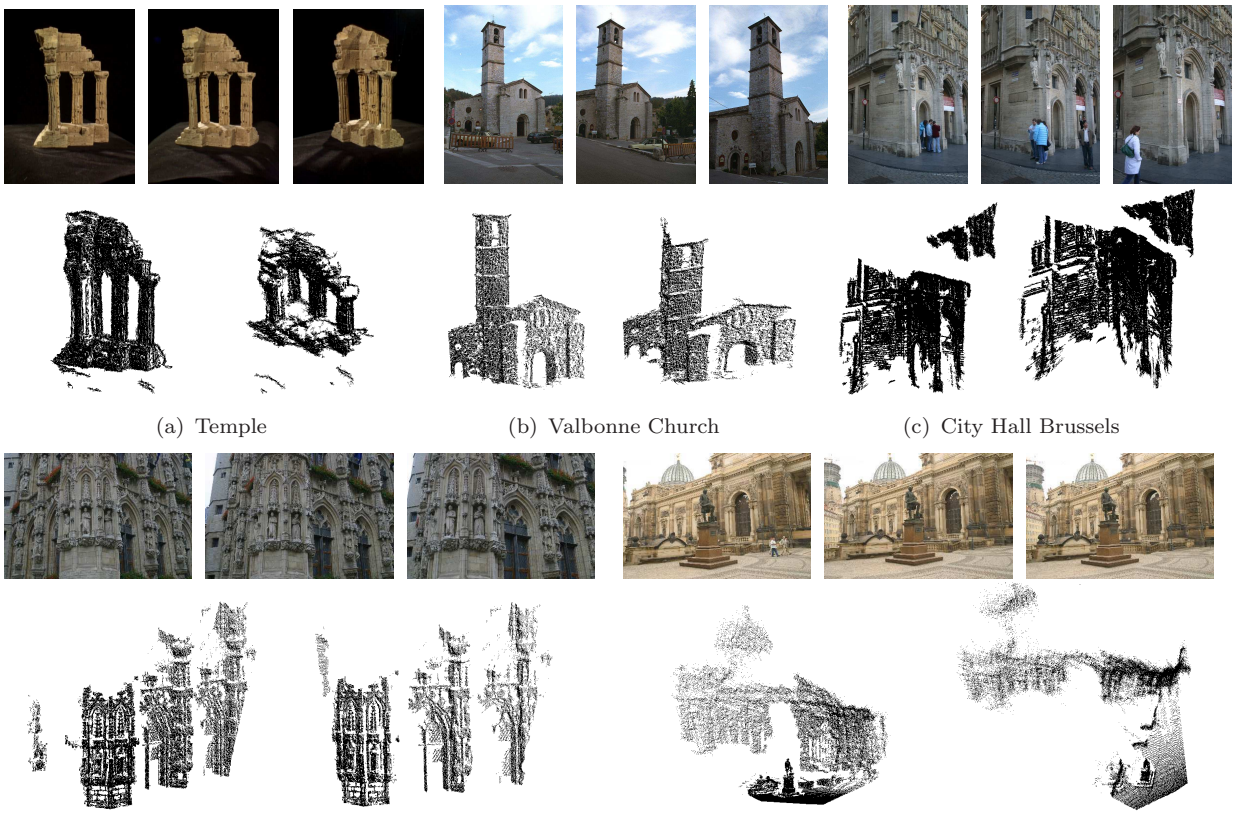
(d) City Hall Leuven



(e) Temple

(f) Shoe

Figure 2.2: **Example output on various datasets.** In each subfigure, the first row shows the input images and the second row shows the corresponding outputs by our method. The bookshelf data set is from J. Matas; City Hall Brussels and City Hall Leuven data set are from C. Strecha [2]; Valbonne Church data set is from the Oxford Visual Geometry Group, while the temple data set is from [3] and the shoe data set is from [4].



(a) Temple (b) Valbonne Church (c) City Hall Brussels

(d) City Hall Leuven (e) Semper Statue Dresden

Figure 2.3: **3D reconstruction from three views.** In each subfigure, the first row contains the three input images and the second row contains two different views of the 3D reconstruction result. Points are shown without texture color for easy visualization of the reconstruction quality. City Hall Brussels, City Hall Leuven, and Semper Statue Dresden data set are from C. Strecha [2]; Valbonne Church data set is from the Oxford Visual Geometry Group, while the temple data set is from [3].

Discussion

In this section, we propose a graph-based semi-supervised symmetric matching framework to perform dense matching between two uncalibrated images. Possible future extensions include more systematic study of the parameters and extension to multi-view stereo. Moreover, we will also pursue a full GPU implementation of our algorithm since we suspect that the current running time is mostly spent on data communication between the CPU and the GPU.

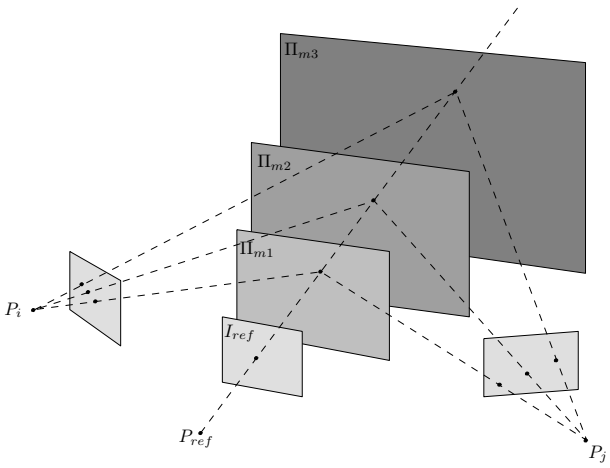


Figure 2.4: **Illustration of multi-view stereo.**

2.2 Stereo matching on GPU

2.2.1 Introduction

In this section, we propose a real-time belief propagation stereo approach. To our knowledge, it is the first real-time multi-view stereo approach based on belief propagation of global optimization. This algorithm is based on a global energy-minimization framework which contains two terms, the data term and smoothness term. Thus our method can be treated as a two-step algorithm: the construction of the data term and the iterative optimization of the smoothness term. The real-time performance is achieved by hardware support Z-culling for fast-converging belief propagation, utilization of temporal coherence information for message initialization, and hardware texture interpolation on GPU. The high-quality result is achieved by render target division, and adaptive depth labeling meaning, with guarantee by belief propagation to achieve MAP configuration.

The inputs to the algorithm are N images at different camera positions and their respective camera projection matrices $P_k = K_k R_k [I| - C_k]$, where K_k is the camera calibration matrix, and R_k, C_k are the rotation and translation of camera P_k with respect to the reference camera P_{ref}. The reference camera is assumed to be the origin of the coordinate system. Accordingly its projection matrix is $P_{ref} = K_{ref} [I_{3\times3}|0]$.

Now, our problem is to compute the depth map for the reference camera. We treat the depth at each pixel as unknown variable, and connect these unknown variables with its four direct neighbors by an undirected edge. Hence, we get a graph of the unknown depth variables and want to get a MAP depth value configuration for this graph. Assuming all probabilities are nonzero, the Hammersley-Cliord theorem (e.g., [1]) guarantees that the probability distribution will factorize into a product of functions of the maximal cliques

of the graph.

Denoting by x the values of all unknown variables in the graph, and each node x_p has a corresponding node y_p that is connected only to x_p. Assuming all the cliques consist of pairs of units, then we have

$$P(x,y) = \prod_{p,q} \Psi_{pq}(x_p, x_q) \prod_{pp} \Psi_{pp}(x_p, y_p).$$

where the first product is over connected pairs of nodes. Assuming the Markov blanket property, we can use the loopy belief propagation scheme used in [108]. At every iteration, each node sends a (different) message to each of its neighbors and receives a message from each neighbor. Let x_p and x_q be two neighboring nodes in the graph. We denote by $m_{pq}(x_p, x_q)$ the message that node x_p sends to node x_q , by $m_{pp}(x_p)$ the message that y_p sends to x_p, and by $b_p(x_p)$ the belief at node x_p. The max-product update rules are

$$m_{pq}(x_q) \leftarrow \alpha \max_{x_p} \left\{ \Psi_{pq}(x_p, x_q)\, m_{pp}(x_p) \prod_{x_k \in N(x_p) \backslash x_q} m_{kp}(x_p) \right\}$$

$$b_p(x_p) \leftarrow \alpha m_{pp}(x_p) \prod_{x_k \in N(x_p)} m_{kp}(x_p)$$

where α denotes a normalization constant and $N(x_p) \backslash x_q$ means all nodes neighboring x_p, except x_q. The procedure is initialized with all message vectors set to constant functions. Observed nodes do not receive messages and they always transmit the same vector – if y_p is observed to have value y^* then $m_{pp}(x_p) = \Psi_{pp}(x_p, y^*)$. The normalization of m_{pq} is not necessary – whether or not the message is normalized, the belief b_p will be identical. However, normalizing the messages avoids numerical underflow and adds to the stability of the algorithm.

2.2.2 Loopy belief propagation on GPU

Instead of computing the max-product, we can do some mathematical manipulation, and compute the min-sum [6]:

$$m_{pq}^t(x_q) = \min_{x_p} \left(V(x_p, x_q) + D_p(x_p) + \sum_{s \in N(p) \backslash q} m_{sp}^{t-1}(x_p) \right)$$

$$b_q(x_q) = D_q(x_q) + \sum_{s \in N(q)} m_{pq}^T(x_q)$$

Now, let $h(x_p) = D_p(x_p) + \sum_{s \in N(p) \setminus q} m_{sp}^{t-1}(x_p)$. As in the traditional two rectified cases, we also assume the smoothness constraint is represented by a truncated linear model:

$$V(x_p, x_q) = \min(s \|x_p - x_q\|, d)$$

where s is the rate of increase in the cost, and d controls when the cost stops increasing. Under this assumption, we can break down the message update as three passes to avoid redundant computation.

Forward Pass

for x_q from 1 to $k-1$:

$m(x_q) \leftarrow \min(m(x_q), m(x_q - 1) + s)$.

Backward Pass

for x_q from $k-2$ to 0:

$m(x_q) \leftarrow \min(m(x_q), m(x_q + 1) + s)$.

Truncated Pass

$m_{pq}(x_q) \leftarrow \min\left(m(x_q), \min_{x_p} h(x_p) + d\right)$.

Z-culling for fast converging

For standard BP algorithms, in order to achieve the best stereo results, a large number of iterations are required to guarantee the convergence. However, since the running time is linear to the number of iterations, large number of iterations will greatly hurt the practical application of BP algorithms.

There are lots of redundant computations involved in standard BP. In essence, by only updating pixels that have not yet converged, fast-converging BP removes those redundant computations while achieving almost the same accuracy as standard BP. So after the pixel shader update the message vectors, we can compare the difference between its new value and its old value from the previous steps. If this difference is too small, we may argue that this pixel is almost converged and stop the further processing for this pixel. This is done by just set the Z-value for this pixel in the pixel shader, to be large enough such that it is outsides the viewing frustum, and hence it is invisible and will not be computed any more.

Render target division

Now, let's consider the problem of encoding all messages sent out from one pixel to its four direct neighbors. Using the multiple render target (MRT) technique, we can render

4 output targets (8 in DirectX 10) with 4 channels for each target. So, totally, we can output 16 float values in each rendering pass. But for each pixel, since we have to store the messages sent to the four directions (up, right, down, left), we can only have 4 float values for each message vector, i.e., we can only have 4 possible depth values.

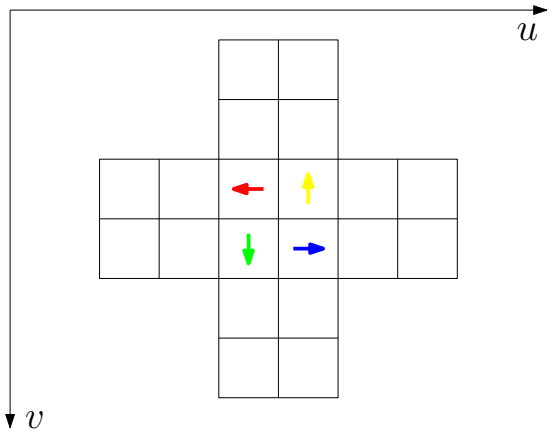


Figure 2.5: **Breaking down the render target.**

Of course, we can perform the same computation several times and for each time, we only output some parts of the message vectors. But in this way, the same computation is wasted. To avoid redundant computation, a key observation is that for the message updating, the message sent to one direction is independent of all the messages sent to the other directions. Hence, instead of restrict one pixel in the reference image to be one pixel in the quad that we render, we can break it down to correspond with 4 pixels in the quad as in Figure 2.2.2. By checking the coordinate, the pixel shader can determine now what direction the message that will be sent to and do the corresponding computation. In such a way, we can handle 16 depth labels. In the new DirectX 10, since we can render 8 target at one pass, totally 32 depth labels are allowed, which is sufficient in most real-time application cases. For more depth labels desired, we can use the technique described in Section 2.2.4.

Temporal coherence for message initialization

One good thing that inherent in belief propagation is that the initialization of the messages can affect the convergence speed and also the final result. For video stereo depth estimation, instead of using coarse-to-fine hierarchical way to initialize the messages which is also bring a lots of overhead, we prefer to utilize the temporal coherence information to initial the messages based on the messages in the previous frame. Assuming the object in the camera move enough slow comparing with the frame frequency, our method in fact can give the almost optimal initial value for the current frame by claiming that the depth

map between two consecutive frames is almost the same. In this way, we only need very few numbers of passes in the pixel shader to achieve satisfactory results.

2.2.3 Fast data cost computation

Before we start the optimization by belief propagation, we have to compute $D_q(x_q)$ first. For a pixel in the image plane with depth d, if we assume the skew of the reference camera is 0, we can get the 3D position of the corresponding point position as:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} (x_{ref} - p_{x_{ref}})\, d/f_{x_{ref}} \\ (y_{ref} - p_{y_{ref}})\, d/f_{y_{ref}} \\ d \end{bmatrix},$$

where $f_{x_{ref}}$ and $f_{y_{ref}}$ is the focal length of P_{ref}, while $p_{x_{ref}}$ and $p_{y_{ref}}$ are the principal point offset. Thus, the location (x_k, y_k) in image I_k of the mapped pixel (x, y) of the reference view is computed by:

$$\begin{bmatrix} x_k\ y_k\ w_k \end{bmatrix}^T = K_k R_k \left[I \right| - C_k \right] \begin{bmatrix} x\ y\ z\ 1 \end{bmatrix}^T.$$

Now, if the plane intersects the surface projected to pixel (x, y) in the reference view, the colors of $I_k\left(\frac{x_k}{w_k}, \frac{y_k}{w_k}\right)$ and $I_{ref}(x, y)$ should be similar assuming Lambertian surfaces. Hence, we can have a likelihood computed from the difference between $I_k\left(\frac{x_k}{w_k}, \frac{y_k}{w_k}\right)$ and $I_{ref}(x, y)$. Here, we use the sum of truncated absolute difference (STAD) between the corresponding pixels:

$$\sum_k \min\left(\frac{1}{3}\sum_{C=R,G,B} \alpha_C \left| I^C_{ref}(x, y) - I^C_k\left(\frac{x_k}{w_k}, \frac{y_k}{w_k}\right)\right|, c_{max}\right)$$

where c_{max} is the truncation value and is set to 33 in the experiment.

However, instead of computing the texture coordinate of each pixel for each depth by the above transformation, a much inexpensive way is to make use of the hardware texture interpolation ability. The idea is just to try to compute the coordinates (x_k, y_k) in the image I_k by this transformation for the only four vertices of the quad that we render as GPGPU technique, and then use the graphics hardware to interpolate the coordinates (x_k, y_k) from the four corner of the quad.

However, since the hardware interpolation is based on the reference camera P_{ref}, but not the camera P_k. So the texture that we look up by just using (x, y) is incorrect (Figure 2.2.3). Here, instead of just storing the (x_k, y_k) for the four corner vertices of

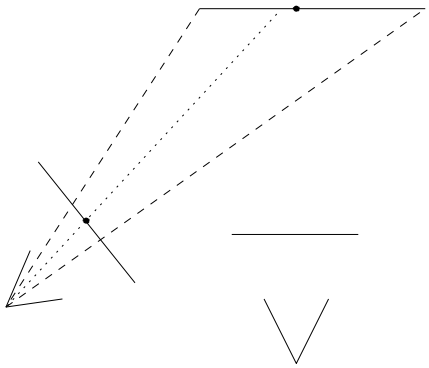


Figure 2.6: **Incorrect hardware interpolation.**

16 labels depths	tsukuba	map	cones	teddy
CPU Implementation[6]	3.4332s	1.7212s	3.5565s	3.7100s
CPU Implementation[7]	1.1100s	0.7000s	1.6900s	1.6300s
Our CPU Method without Hierarchy (30 iter)	2.6653s	1.4504s	3.8475s	3.8624s
Our CPU Method with Hierarchy (4 iter 5 level)	0.4354s	0.2520s	0.6775s	0.6680s
Our GPU Method without Hierarchy (30 iter)	0.0155s	0.0154s	0.0158s	0.0156s
Our GPU Method with Hierarchy (4 iter 5 level)	0.0034s	0.0034s	0.0033s	0.0033s

Table 2.1: **Time comparison for the results in Figure 2.2.5.**

the quad, we store the whole (x_k, y_k, w_k) and then interpolate the whole triple with w_k. Then for each pixel in the pixel shader, instead of doing texture lookup with interpolated coordinate (x, y), we use the coordinate $\left(\frac{x}{w}, \frac{y}{w}\right)$ which will get the correct result.

2.2.4 Adaptive labeling depth

As mention before, in order to be able to update the messages without redundant computation, even with MRT technique, we can only handle up to 16 depth labels. In order to produce the high quality result, we have to increase the possible number of depth labels. The idea is to change the meaning of the depth labels. Rather than force each label for each pixel has the same depth meaning, we can variate the depth meaning for different pixel.

There is two catalogs to handle this. One is to use a coarse-to-find approach to first use all depth labels to span the whole depth interval $[d_{near}, d_{far}]$. Then after several iterations, say j, we can find the best one, and then shrink the spanning interval to be a smaller one centered at the current best depth estimation. This idea can be further extends to let the spanning interval to be divided into i intervals with each one center at the top i depth estimation in the previous estimation. In particular, when $j = 1 = i$, we update the interval each pass adaptive according to the current depth estimation.

The other way is for video with temporal coherence, we can just make the depth labels spanning a small range of depths centered at the depth estimated in the previous video frame. But for the object boundary, this method is incorrect since the depth can change dramatically. To handle this, instead of using one depth interval centered at the depth estimated for the same pixel in the previous video frame, we can use several depth intervals centered at the depths estimated in the previous video frame for not just the same pixel, but also the neighbor pixels. By using multiple possible depth intervals, the problem is solved.

These two ways is not exclusive and can be used together to further refine the result with only limit number of possible labels. In term of implementation, in order to know the actual meaning of the depth labels, we have to record the actual depth of the depth labels, correspondingly compute, in the pixel shader, the parameter s in the truncated linear model for smoothness constraint on the fly.

2.2.5 Experiments

We tested our algorithm on a 2.13 GHz Dual-core PC with 1GB RAM. The GPU is a Geforce 8800 GTX graphics card with 512M video memory from NVIDIA. All shaders are implemented using Cg with OpenGL. The following experiments are conducted to evaluate both the quality and efficiency performance of our algorithm. Figure 2.2.5 compare the result of two view stereo with the other two implementations. And the time to generate these results is given in Table 2.2.3.

Cameras Synchronization Since our method is very fast, we can use it to synchronize different cameras. For instance, we have four Point Grey cameras and two PC. So each PC has two cameras mounted. For the two cameras in the same PC, we can easily synchronize it by software API. In our cases, we use Point Grey MultiSync software to synchronize the two cameras in the same PC. Now, the problem is to synchronize the camera between two different PCs. In stead of using IEEE1394 hardware signal to synchronize, our approach is first to record the videos separately. Since we have synchronized the two cameras in the same PC, we can use our proposed method to compute the depth from two pairs of cameras. After that, we have two depth map videos, and we can use brute force search to find the best match and thus synchronize them.

Discussions In this section, we address multi-view stereo under a global optimization approach based on belief propagation while maintaining real-time performance. To our knowledge, it is the first real-time multi-view stereo approach based on belief propagation

of global optimization. Also, based on the high-speed computation of our methods, a novel application to our method for automatic camera synchronization is proposed. Real world examples demonstrate the efficiency of our method. Possible future extensions include the optimization together with the temporal smoothness constraints. Our progressing work is to, rather than frame-by-frame, construct a 3D grid graph together with time-axis to achieve the global optimization for the full video sequences.

2.3 Sequence merging strategy

After achieving a set of point correspondences between image pairs, we now need to divide the whole set of images into subsequences for reconstruction, and merge the reconstruction results from subsequences while performing metric upgrade at some time. At the same time, it's not a good idea to compute all the camera locations and use the bundle adjustment only once on the whole sequence. In that case, increasing errors could produce an initial solution too far from the optimal one for the bundle adjustment to converge. Thus it is necessary to use the bundle adjustment throughout the reconstruction of the sequence.

We argue that different strategies should be used for different input data. If the objective of the reconstruction is to model a specific object, we may take photos in a circular movement towards that object. To guarantee the robustness, we use a hierarchical method for merging. Or, if we cannot control the image capture procedure, and only get a set of unstructured data, we have to use more general method. For other situations, such as street-view city modeling, even the data capture is well controlled, we still cannot afford the computation cost of the hierarchical method, and a incremental method with local bundle adjustment will be used.

2.3.1 Strip sequence with hierarchical merging

A strip sequence is an image sequence with known single strip order such that the camera positions of two consecutive images are most close. A circular sequence is a special case. In our method, the strip sequence $(1, 2, \ldots, n)$ is divided into two parts with an overlap of two frames in order to be able to merge the sequence, i.e. $\left(1, \ldots, \left\lfloor \frac{n}{2} \right\rfloor, \left\lfloor \frac{n}{2} \right\rfloor + 1\right)$ and $\left(\left\lfloor \frac{n}{2} \right\rfloor, \left\lfloor \frac{n}{2} \right\rfloor + 1, \ldots, n\right)$. Each subsequence is recursively divided in the same way until each final subsequence contains only three images.

Now in each image triplet, taking the middle image in the triplet, the i-th image, two pairs of consecutive images, $(i, i-1)$ and $(i, i+1)$ are matched as described in Section

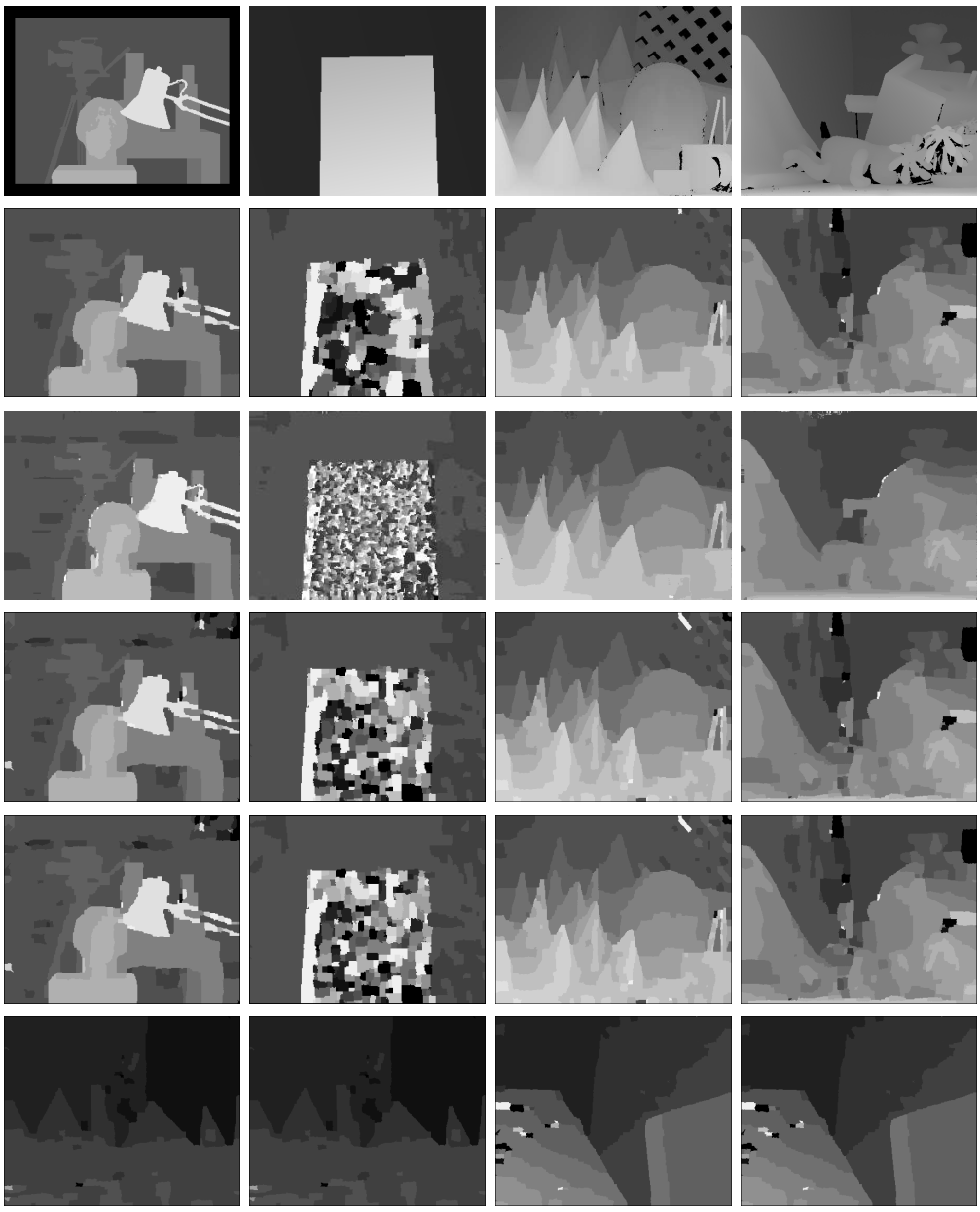


Figure 2.7: **Result comparison on Middlebury dataset [5].** The first row is the ground truth. The second row is produced by the source codes provided by [6]. The third row is produced by the donated executable file from Qingxiong Yang [7]. The fourth row is the result of our CPU method. The fifth row is the result of our GPU method. The last row is our result of the other two datasets with CPU version on the left and GPU version on the right. The corresponding running times are listed in Table 2.2.3.

2.1. Using the i-th image as the bridge, we get a set of point correspondence tracks across the three images. Now, the trifocal tensor $\mathcal{T}$ can be estimated by the Gold Standard algorithm[90] under a RANSAC framework followed by a bundle adjustment to minimize the projection errors.

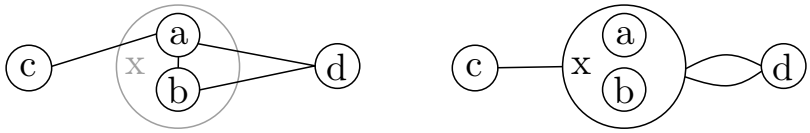
In order to merge two sequences $\left(\ldots, \left\lfloor \frac{n}{2} \right\rfloor, \left\lfloor \frac{n}{2} \right\rfloor + 1\right)$ and $\left(\left\lfloor \frac{n}{2} \right\rfloor, \left\lfloor \frac{n}{2} \right\rfloor + 1, \ldots\right)$, we use the last 2 cameras $\left\lfloor \frac{n}{2} \right\rfloor$ and $\left\lfloor \frac{n}{2} \right\rfloor + 1$. As the images are the same, the cameras associated after merging must be the same. Suppose a point has coordinate $\mathbf{X}_i$ in the first sequence and $\mathbf{X}'_i$ in the second. If all measurements were perfect, then there would exist a homography H of 3-space between the two projective frames such that $P_j = P'_j H^{-1}$ and $\mathbf{X}_i = H\mathbf{X}'_i$, where P_j, P'_j are the corresponding camera matrices for images common to the subsequences. With real image sequences the relationship will not obeyed exactly, and the homography is estimated with least square. Then, a bundle adjustment is used on the result of the merging operation. If the subsequence is metric, instead of estimating the homography, we estimate a similarity transformation which includes rotation, scale and translation. Merging is done until the whole sequence has been reconstructed. For close circular sequence, we do the matching between the first and the last image, and add the reconstruction points into the sequence. Finally, the reconstruction ends with a global bundle adjustment.

For a short sequence, we perform all projective reconstruction and after achieved a global bundle adjusted reconstruction, we upgrade it to metric. For a longer sequence, this method may not good due to the accumulated error in the projective reconstruction. Hence, if a subsequence has more than 60 images, we metric upgrade it first before further merging.

2.3.2 Unstructured sequence with clustering based merging

If we cannot control the image capture procedure, and only get a set of unstructured data, we have to use more general method. In [76], an initial image pair is selected and the camera parameters for this pair are estimated using the five point algorithm, then the tracks visible in the two images are triangulated with a two frame bundle adjustment starting from this initialization. Then, they add one camera at a time to the optimization in an incremental manner. Hence, their method is very sensitive to the error in the initial frame. Instead, we still want to use a hierarchical merging method that is more robust. Since the image sequence is in unorganized way, a more general graph clustering based merging method is proposed.

Using the correspondences between all image pairs, we can construct an image con-



(a) Before a and b is merge. (b) After a and b is merge.

Figure 2.8: **Node contraction.**

nectivity graph, in which each image is a node and an undirected edge exists between any pair of images with enough matching correspondences. The weights on the edges between the image pairs are defined as the multiplication of image-based distance computed in the previous section and the square root of the number of point correspondences. Hence, if the number of point correspondences is large and the transformation is not too degenerate to a planar homography, the weight will be large.

Now, we hierarchically cluster the nodes in the graph. For each step, we choose the edge with the largest weight, corresponding to the pair of images with largest image-based distance, and merge the two nodes. These two nodes will be contracted into a single node in the graph. All edges incident to any of these two nodes will be incident to the new node from contraction.

Hence, there are two types of nodes: nodes representing a single image, and nodes representing a subsequence. At the beginning of the hierarchical merging operation, each node represents one single image. After merging, the contracted node represents a subsequence. If two image nodes are merged, the fundamental matrix will be estimated to achieve a projective reconstruction. If an image node and a subsequence node is merged, the point correspondences between the single image and all images from the subsequence will be used to estimated the camera matrix of the image node by the DLT method [90] inside a RANSAC framework. If two subsequence nodes are merged, the point correspondence tracks are first merged, and then the homograph transformation is estimated as in Section 2.3.1. In all three kinds of merging, a bundle adjustment is used to refine the merging results. As before, if a merging result is a subsequence containing more than 60 images, it is metric upgraded with an extra bundle adjustment.

In detail, as shown in Figure 2.3.2, supposed two nodes a and b are contracted into a single node x, and there are some edges (a, c), (a, d) and (b, d). Since b and c is not connected, there will be one edge (x, c) after merging. But since both a and b are connected with d, the situation is complicated. Because we want to try our best to preserve the relation between (a, d) and (b, d)in order to improve the robustness if later we merge x and d, these two edges should both be preserved. And there will be two edges between x and d. If later, any one of the two edges between x and d is found to have

the largest weight in the graph, x and d will be merged, and all edges between x and d will be used to compute the image correspondence tracks. If the edge between x and d is used due to its largest weight value, it must have the largest weight value among all edges between x and d. Hence, in practice, we don't need to store the two edges explicitly but just store the largest weight edge and merge the information of the other edges into this largest weight edges.

Our method is naturally adoptable for acceleration of large sequence reconstruction. In order to speed up for large sequence merging, the bundle adjustment strategy can be different. Assume now we want to merge a large subsequence with another subsequence or image, instead of doing bundle adjustment with all camera parameters and all points' 3D coordinates, we just do the bundle adjustment on the transformation parameters, i.e. the homography between one projective subsequence with another (metric or projective) subsequence, the camera parameters for an image node, or the similarity transformation between two metric subsequences.

2.3.3 Long sequence with incremental merging

For very long sequence such as in the application for street-view city modeling, the above two strategies is still too slow for computation. For such a long sequence, any global bundle adjustment should be avoided. Hence, we use an incremental method with local bundle adjustment only, and add in one camera at a time. We begin by estimating a short sequence at the very beginning with the method in Section 2.3.1. Next, we add another camera to the optimization, and initialize the new camera's parameters using DLT technique inside a RANSAC procedure with local bundle adjustment in the last few frames as in [109].

To improve robustness, we make a few modifications to the basic procedure outlined above. After every bundle adjustment run, we detect outlier tracks that contain at least one keypoint with high reprojection error, and remove these tracks. We then rerun the bundle adjustment, rejecting outliers after each run, until no more outliers are detected.

If a subsequence is metric upgraded, we also reject a point if it has projection observation on a particular camera and its 3D position is behind that camera. This can be checked by the dot product between the camera orientation vector and the vector between the point and the camera position. Finally, we reject the points too far away form the center of the point cloud, such as the point for the cloud in the sky. Although these very far away points may be at correct position, they are sensitive to noise and their existence will drift the whole sequence into bad local minimal during bundle adjustment. For each

image, we also reject a point with projection in it, if there exist several nearby points (from different directions in the image) with projection very close to the projection of that point, and are very far away from that point in 3D space. After the outlier rejection, the bundle adjustment is rerun.

CHAPTER 3

SEGMENTATION

The objective of segmentation is to isolate the target data for the further processing of modeling. Depended on the available information, different algorithms are used. When no training data is available, clustering and grouping approach is the choice. With these clustering approaches, we want to jointly utilize the 2D image and 3D geometry information for segmentation. On the other hand, if training data is available, it is possible for us to obtain some semantically meaningful segmentation. In this chapter, we first introduce some algorithms and analysis for clustering and grouping in Section 3.1. Then, a joint 2D and 3D segmentation is introduced in Section 3.2. In Section 3.3, we introduce a semantic segmentation that can identify different kinds of object classes for building modeling.

3.1 Clustering and grouping

3.1.1 Affinity propagation

Introduction

Recently, Affinity Propagation (AP) [110, 111] has become one of the most powerful and popular methods for unsupervised clustering. Since many promising results have been found on various data sets and applications [111, 110, 112, 113, 114, 115, 116], great attention rises in many fields of both scientific research and industry. However, most of the following works focus on extension of AP [110, 111] to various kinds of applications, while the updated rules for AP have not yet been theoretically justified. Very little work has been done on understanding the reason why AP works, the relation with previous methods, and the limitation of AP. In [111, 110], though an interpretation by factor graph is provided, the procedure is still based on qualitative reasoning.

In this work, we take an early step to show the great similarity between AP and the Expectation Maximization (EM) [117] for fixed-parameter Mixture Model (MM). With side-by-side analysis, we found that the availabilities in AP have great similarity with the component weights in Mixture Model, and the automatic determination of the number of the clusters in AP imitates the estimation of the component weights for M-step in EM. Our interpretation of the preferences in AP lends to the discovery that the exemplar of AP

may not be the exemplar for itself, which means the objective function in [111] is actually not the real one for AP. On the other hand, due to the major dissimilarity between AP and MM-EM, i.e. the absence of a component weight term in responsibility update rules, AP without damping suffers from vibration in every two iterations and does not converge. Adding the term back lends to the normalized AP that empirically converges without damping. Various experiments demonstrate and evaluate our observations and analysis.

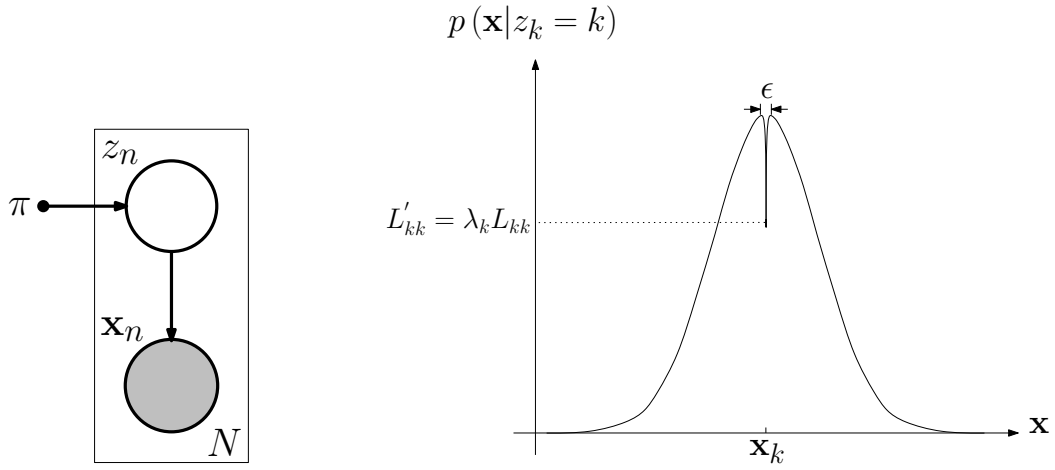
Analysis

Review and notation Affinity Propagation (AP) takes as input a collection of real-valued similarities between a set of d dimensional data point $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, where the similarity L_{ik} indicates how well the data point k is suited to be the exemplar for data point i. Normally, we use a Gaussian likelihood

$$L_{ik} = \mathcal{N}\left(\mathbf{x}_i | \, \mathbf{x}_k, \mathbf{\Sigma}_k\right) = \frac{1}{(2\pi)^{d/2} \left|\mathbf{\Sigma}_k\right|^{1/2}} \exp\left\{-\frac{1}{2}\left(\mathbf{x}_i - \mathbf{x}_k\right)^T \mathbf{\Sigma}_k^{-1}\left(\mathbf{x}_i - \mathbf{x}_k\right)\right\}$$

where $\mathbf{\Sigma}_k = \mathbf{\Sigma}$ for $k = 1, \ldots, N$, and $\mathbf{\Sigma}$ is a fixed $d \times d$ covariance matrix set by users. This likelihood function is suggested by sum-product AP in [110]. For the max-sum AP in [111], since **log** probability is used to compute the responsibility as shown in Section 3.1.1, the likelihood is defined as the exponent $s_{ik} = -\frac{1}{2}\left(\mathbf{x}_i - \mathbf{x}_k\right)^T \mathbf{\Sigma}^{-1}\left(\mathbf{x}_i - \mathbf{x}_k\right)$, or degenerates to negative Euclidean distance $s_{ik} = -\left\|\mathbf{x}_i - \mathbf{x}_k\right\|^2$ as suggested in [111] when $\frac{1}{2}\mathbf{\Sigma}^{-1} = \mathbf{I}$. Other kinds of similarity measure can also be used. As we will see in Section 3.1.1, in fact, the similarity measure decides the probability density function of the models to be mixed. In AP, the likelihood needs to be fixed beforehand by providing the similarity matrix $\mathbf{L} = [L_{ik}]$ (or equivalently $\mathbf{S} = [s_{ik}]$). For L_{kk}, AP requests a special **preference** value that influences the result number of clusters. To distinguish with $L_{kk} = \mathcal{N}\left(\mathbf{x}_k | \, \mathbf{x}_k, \mathbf{\Sigma}_k\right)$, we denote the preference values as L'_{kk}, and define the ratio $\lambda_k = L'_{kk}/L_{kk}$.

Side-by-side analysis As shown in Figure 3.1(a), let $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ be the observed random variables, and $\mathbf{Z} = \{z_1, \ldots, z_N\}$ be the latent random variables. The great similarity between AP and GMM-EM occurs when there are N components in GMM-EM, i.e. the same as the number of observed variables, with mixing coefficients $\boldsymbol{\pi} = \{\pi_1, \ldots, \pi_N\}$, while the k-th Gaussian component has the mean at the position $\mathbf{x}_k$ of the k-th data point with the covariance matrices $\mathbf{\Sigma}_k$. Both $\mathbf{x}_k$ and $\mathbf{\Sigma}_k$ are fixed and known for the k-th component, and π_k is an unknown fixed point to be estimated. For such a GMM with



(a) Fixed-parameter mixture model. (b) Gaussian distribution with infinitesimal indent.

Figure 3.1: **Mixture model and component distribution profile.**

fixed mean and covariance matrix,

$$p(\mathbf{x}|\boldsymbol{\pi}) = \sum_{k=1}^{N} \pi_k p(\mathbf{x}|z_k = k) = \sum_{k=1}^{N} \pi_k \mathcal{N}(\mathbf{x}|\mathbf{x}_k, \boldsymbol{\Sigma}_k),$$

we can use the standard EM method [117].

E-step In EM, the E-step is to estimate the unobserved $\mathbf{z}$, conditioned on the observation, using the values from the last M-step:

$$\widetilde{r}_{ik} = p\left(z_i = k|\mathbf{x}_i, \pi_k^t\right) = \frac{\pi_k^t p(\mathbf{x}_i|z_i = k)}{\sum_{j=1}^{N} \pi_j^t p(\mathbf{x}_i|z_i = j)} = \frac{\pi_k^t L_{ik}}{\sum_{j=1}^{N} \pi_j^t L_{ij}}. \tag{3.1}$$

In AP, this corresponds to the responsibility r_{ik}, sent from data point i to candidate exemplar point k, reflects the accumulated evidence for how well-suited point k to serve as the exemplar for point i, taking into account other potential exemplars for point i. When $i \neq k$, the updated rule in sum-product AP [110] is:

$$r_{ik} = \frac{L_{ik}}{\sum_{j=1,j\neq k}^{N} a_{ij} L_{ij}} = \frac{L_{ik}}{\sum_{j=1}^{N} a_{ij} L_{ij} - a_{ik} L_{ik}}, \tag{3.2}$$

which is very similar with Eq. 3.1. (We will discuss the absence of component weight π_k^t in the numerator in Section 3.1.1, and exclusion of $j = k$ in the sum in Section 3.1.1). When $i = k$,

$$r_{kk} = \frac{L'_{kk}}{\sum_{j=1,j\neq k}^{N} a_{ij} L_{ij}} = \frac{\lambda_k L_{kk}}{\sum_{j=1,j\neq k}^{N} a_{ij} L_{ij}}. \tag{3.3}$$

Note that λ_k is a factor in the numerator with the likelihood L_{kk}. Preserving the relative order, the max-sum updated rule in [111] is $r_{ik} = s_{ik} - \max_{j \neq k} \{a_{ij} + s_{ij}\}$ where $s_{ik} = \log L_{ik} + C$ for $i \neq k$, and $s_{kk} = \log L'_{kk} + C$ for $i = k$. Here, $C = \log((2\pi)^{d/2} |\mathbf{\Sigma}|^{1/2})$ is a constant that will be canceled out by the subtraction. There are two major modifications from sum-product form [110] to max-sum form [111]: First, the **product** to **sum** conversion transfers the computation from probability domain to **log** probability domain. Hence, the likelihood becomes $s_{ik} = -\frac{1}{2} (\mathbf{x}_i - \mathbf{x}_k)^T \mathbf{\Sigma}^{-1} (\mathbf{x}_i - \mathbf{x}_k)$. Second, the **max** operation is to exaggerate the effect for the most likely exemplar and suppress the unlikely ones. Therefore, the responsibility is more similar with k-means and k-centroids to give harder assignment to each data point, while Mixture Model gives softer assignment. Note that a_{ij} in $r_{ik} = s_{ik} - \max_{j \neq k} \{a_{ij} + s_{ij}\}$ should be $\log a_{ij}$. However, since the relative order between all a_{ij} is still preserved in all $\log a_{ij}$, two versions of AP [110, 111] do not distinguish them.

M-step In EM, the M-step is to maximize the expected log-likelihood of the joint event with Lagrange multiplier. Since we have fixed mean and covariance matrix for GMM, the updated rule is

$$\pi_k^{t+1} = \frac{1}{N} \sum_{i=1}^{N} p\left(z_i = k | \mathbf{x}_i, \pi_k^t\right) = \frac{1}{N} \sum_{i=1}^{N} \widetilde{r}_{ik}. \tag{3.4}$$

In AP, the availability a_{ik}, sent from candidate exemplar point k to point i, reflects the accumulated evidence for how appropriate it would be for point i to choose point k as its exemplar, taking into account the support from other points that point k should be an exemplar. When $i \neq k$, the updated rule in [111] is

$$a_{ik} = \min \left\{ 0, r_{kk} + \sum_{j=1, j \neq i, j \neq k}^{N} \max \{0, r_{jk}\} \right\}. \tag{3.5}$$

The **min** and **max** operators are to truncate the cost in order to magnify the effect because it is only necessary for a good exemplar to explain some data points well, regardless of how poorly it explains other data points [111]. And this truncation is only used for max-sum AP that operations with log-probability, and does not exist for sum-product AP [110]. If we remove it (see Section 3.1.1 for the effect of the removal), the availability becomes

$$a_{ik} = r_{kk} + \sum_{j=1, j \neq i, j \neq k}^{N} r_{jk} = \sum_{j=1, j \neq i}^{N} r_{jk}.$$

The same as in the E-step, if we also consider i in the contribution of effort (see Section 3.1.1 for the effect of the exclusion), it becomes

$$a_{\cdot k} = \sum_{j=1}^{N} r_{jk}, \tag{3.6}$$

which turns out to be an unnormalized version of update rule for component weight π_k of EM in Eq. 3.4 (the normalization factor $\frac{1}{N}$ will be discussed in Section 3.1.1.). Similarly, when $i = k$, the updated rule $a_{kk} = \sum_{j=1, j\neq k}^{N} \max\{0, r_{jk}\}$ in [111] is an unnormalized and truncated version of the update rule of EM in Eq. 3.4.

Normalization Comparing Eq. 3.6 and Eq. 3.4, the unnormalization will let $a_{\cdot k}^t \approx N\pi_k^t$. This issue will not cause any problem in EM since Eq. 3.1 will have a factor N in both the numerator and denominator. In sum-product responsibility AP, the numerator in Eq. 3.2 does not have the factor to cancel out N with the denominator. With this N times larger $a_{\cdot k}^t$, r_{ik}^{t+1} will be N times smaller. And then $a_{\cdot k}^{t+1} = \sum_{j=1}^{N} r_{jk}^{t+1}$ will be back to the correct magnitude $a_{\cdot k}^{t+1} \approx \pi_k^{t+1}$ while r_{ik}^{t+2} will also be at correct magnitude $r_{ik}^{t+2} \approx \widetilde{r}_{ik}^{t+2}$. Now, once more, $a_{\cdot k}^{t+2}$ will be N times larger than π_k^{t+2}, and vibration continuous in every two iterations. For max-sum responsibility AP, the same situation happens in every two iterations, but with more complex relation, since the log-probability is used for responsibility while probability is used for availability. In Section 3.1.1, the experiments to track the value for $|a_{ik}^t|$ demonstrate our observation. The vibration is in fact experimental observed in [110, 111]. However, it can not be well explained, and hence a damping factor is introduced in [110, 111] to smooth the vibration between two consecutive iterations. In sum-product responsibility AP, with the damping factor, the vibration may converge into a situation when $a_{\cdot k}^t \approx \sqrt{N}\pi_k^t$ and $r_{ik}^t \approx \widetilde{r}_{ik}^t/\sqrt{N}$.

We modified AP to add a_{ik} as the first term of responsibility update rule for max-sum AP (or in the numerator for sum-product AP), called normalized AP, shown as [+n Max-sum AP] in Table 3.1.1, and get less vibration in two consecutive iterations with faster convergence as shown in Section 3.1.1. Hence, we believe that, the major dissimilarity between AP and GMM-EM, the absence of a_{ik} in the numerator of Eq. 3.2 and 3.3, is a disadvantage of AP compared with GMM-EM.

As point out in [118], theoretical guarantees on the convergence or optimality of AP are not yet established. However, in practice, AP usually converges to some reasonable results [110, 111, 112, 113, 118, 114, 115]. Because of its great similarity with EM, we believe that this is due to the convergence of EM. And because EM only converges to local optimum, AP might also converge to local optimum.

Availability and component weight Shown in Table 3.1.1, with some modifications that we have discussed, AP has the exact same form as GMM-EM. In fact, the responsibility of AP is directly borrowed from K-medoids, and hence should be very similar with MM-EM as expected.

The novelty part of AP is to introduce the availability a_{ik} to reflect the accumulated evidence for how appropriate it would be for point i to choose point k as its exemplar, taking into account the support from other points that point k should be an exemplar. This part is missing in the K-medoids method that inspires the original work. However, this idea turns out to be another viewpoint of the M-step in EM to re-estimate the component weight π_k, while the updating procedure for the component weights can be regarded as automatic determination for the number of clusters.

In EM, component weight π_k^0 is usually initialized to be 1. Although theoretically we should impose the constraint $\sum_{k=1}^{N} \pi_k^0 = 1$, the first E-step of EM will scale π_k^0 down uniformly. Hence, we do not need to explicitly normalize the initial $\boldsymbol{\pi}^0$. Correspondingly in sum-product AP [110], a_{ik} is also initialized to be 1. For max-sum AP, as discussed in 3.1.1, a_{ik} in fact represents $\log a_{ik}$. Hence, it is suggested to be initialized as $\log 1 = 0$ in [111]. On the other hand, as an acceleration scheme for implementation, when the weight of particular component is close to zero, i.e. $\pi_k < \epsilon$ or $a_{\cdot k} < \epsilon$, it will never become large again. Hence, we can safely prune that component without introducing any artifact.

Assignment identification In AP, for point i, the value of

$$c_i = \arg\max_k \{a_{ik} + r_{ik}\} \tag{3.7}$$

either identifies point i as an exemplar if $c_i = i$, or identifies the data point that is the exemplar for point i. Since, $a_{ik} + r_{ik} \approx \sqrt{N}\pi_k + \widetilde{r}_{ik}/\sqrt{N}$ for sum-product responsibility AP,

$$c_i \approx \arg\max_k \{N\pi_k + \widetilde{r}_{ik}\}.$$

In GMM-EM, directly based on $\widetilde{r}_{ik}$, the exemplar c_i for point i is from

$$c_i = \arg\max_k \{\widetilde{r}_{ik}\}.$$

However, when $\widetilde{r}_{ik}$ is large for some k, π_k should also be large, and when $\widetilde{r}_{ik}$ is small, π_k should also be small. Hence, the AP exemplar identification is similar with the one in EM. Experiments in Section 3.1.1 shows that similar results are found with these two kinds of assignment identification rules.

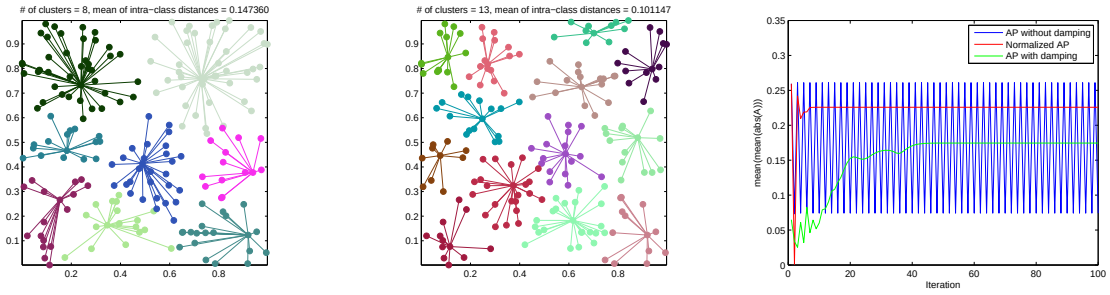
Preference and objective function One of the strongest advantages of AP is that it can automatically adjust the number of clusters, which is similar to adjust the component weight π_k in order to prune out some insignificant components. [110, 111] concludes that the number of result clusters is controlled by L'_{kk}. This is because that AP requires to specifically scale $L_{kk} = p(\mathbf{x}_k|z_k = k)$ by the factor λ_k in Eq. 3.3 through the setting of L'_{kk}. In [111], the preference L'_{kk} is suggested to be empirically set as the median of the input similarities (resulting in a moderate number of clusters) or their minimum (resulting in a small number of clusters). Therefore, L'_{kk} is set to satisfying $\lambda_k = \frac{L'_{kk}}{L_{kk}} < 1$. By explicitly pulling down the value $L_{kk} = p(\mathbf{x}_k|z_k = k)$, the responsibility r_{kk} is smaller than it should be. Hence, the k-th component with mean at data point k needs more support from other data points in order to have a significant weight (availability). By tuning L'_{kk} or λ_k, the number of clusters can be controlled. More systematically, we can view this modification as to let the Gaussian distribution have an infinitesimal-width gap at the mean as in Figure 3.1(b). However, AP does not use this profile as in Figure 3.1(b), but recognizes the point index. Therefore, if two points are at the same position, it is highly possible to let them be the exemplars of each other.

Conversely, in the supporting material for [111], during the derivation of AP as the max-sum algorithm in a factor graph, AP is stated to maximize the objective function

$$O(\mathbf{c}) = \sum_{i=1}^{N} s_{ic_i} + \sum_{k=1}^{N} \delta_k(\mathbf{c}) \tag{3.8}$$

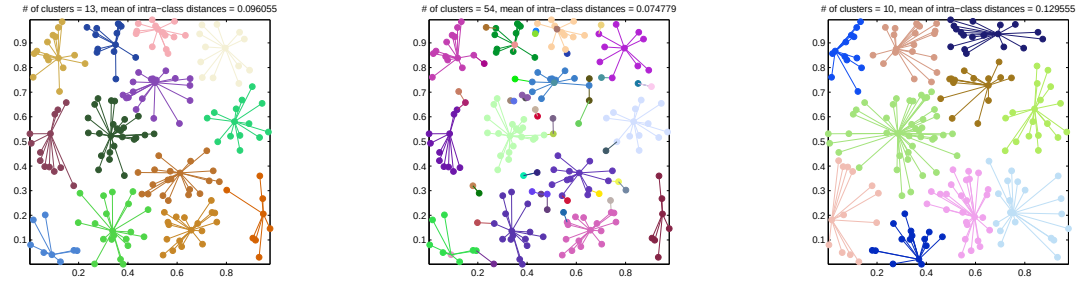
where $\mathbf{c} = (c_1, c_2, \ldots, c_N)$ and $\delta_k(\mathbf{c}) = -\infty$ if $c_k \neq k$ but $\exists i$: $c_i = k$. Hence, every exemplar should be the exemplar of itself. Otherwise, the value for the objective function is $-\infty$. This objective function directly conflicts with our analysis. In Section 3.1.1, three examples report that some exemplars are not the exemplar of themselves. Hence, conflicted with the arguments in [111, 113], we conclude that the objective function in Eq. 3.8 is actually not the one for AP to optimize, and AP cannot guarantee $\delta_k(\mathbf{c}) \neq -\infty$ with the result extraction rule $c_i = \arg\max_k \{a_{ik} + r_{ik}\}$ from [111].

Limitation and extensions With the above analysis, one key limitation of AP is that the parameters for the parametric models used for mixing, such as mean and covariance matrix in Gaussian case, should be fixed and known. Since there are the same number of components with data points, fixing the mean for each component is a reasonable assumption, but fixing the covariance matrix is not. In EM, this is handled in the M-step to re-estimate the covariance matrix for the k-th component. Correspondingly in AP, with $r^t_{ik} \propto \widetilde{r}^t_{ik}$, we can have $\Sigma^{t+1}_k = \sum_{i=1}^{N} r^t_{ik} (\mathbf{x}_i - \mathbf{x}_k)(\mathbf{x}_i - \mathbf{x}_k)^T / \sum_{i=1}^{N} r^t_{ik}$.



(a) Normalized AP (b) AP with damping=0.5 (c) Mean of absolute availabilities

Figure 3.2: **Vibration and normalization.** AP without damping cannot produce clustering result by the original implementation[8]. And normalized AP does not need any damping factor in order to converge. Refer to [+n Max-sum AP] in Table 3.1.1 for update rules of normalized AP.



(a) $k^* = \arg\max_k \{a_{ik} + r_{ik}\}$ (b) $k^* = \arg\max_k \{r_{ik}\}$ (c) $k^* = \arg\max_k \{r_{ik}\}$

Figure 3.3: **Component assignment identification.** (a) and (b) are the original max–sum AP with damping factor=0.5, (c) is the normalized max-sum AP without damping. The assignment identification rules are shown as subfigure title.

Empirical Study

Differences between AP and GMM-EM In these experiments, 200 2D random points in $[0,1] \times [0,1]$ are generated uniformly. And we empirically evaluate several differences between AP an GMM-EM that we have discussed. Various message update rules are listed in Table 3.1.1.

Normalization Discussed in Section 3.1.1, the major difference between AP and MM-EM is the normalization issue. Experimentally, shown in Figure 3.1.1, we trace the mean of absolute availabilities in 100 iterations.

Assignment identification In Section 3.1.1, we compare two assignment identification rules: $c_i = \arg\max_k \{a_{ik} + r_{ik}\}$ and $c_i = \arg\max_k \{r_{ik}\}$. Experimentally, shown in Figure 3.1.1, we use these two rules on max-sum AP. Furthermore, we use $c_i = \arg\max_k \{r_{ik}\}$ for normalized AP ([+n Max-sum AP] in Table 3.1.1). We can see that the clustering results from both max-sum AP with $c_i = \arg\max_k \{a_{ik} + r_{ik}\}$ and normalized AP with $c_i = \arg\max_k \{r_{ik}\}$ are quite similar.

Exclusion and truncation During the analysis in Section 3.1.1, we suspect that the exclusion of target points in the message and the truncation with 0 does not matter much for AP. One example is given in Figure 3.1.1 where similar results are obtained, while exclusion and truncation are disabled respectively.

Objective function and hard constraint assignment We use several toy examples to verify our conclusion in Section 3.1.1 that AP cannot always guarantee each exemplar to be the exemplar of itself. Note that we exactly follow the paper in [111] to identify the exemplar by $c_i = \arg\max_k \{a_{ik} + r_{ik}\}$, since the implementation provided in [8] may have some other procedures to guarantee the hard assignment.

Example 1 There are three data points in this example. We use the ultra-short Matlab script for max-sum AP from [8] and the support material for [111], and let max-sum AP to run 10000 iterations. In Matlab, we use "[value,idx] = max(E,[],2)" for $c_i = \arg\max_k \{a_{ik} + r_{ik}\}$ instead of "I=find(diag(E) > 0); K=length(I); [tmp c]=max(S(:,I),[],2); c(I)=1:K; idx=I(c);" provided by [8] to extract results. We use non-metric distance that AP claimed to support in [112, 111, 110]. The similarity matrix $\mathbf{S}$ used for computation and the result extraction matrix $\mathbf{E} = \mathbf{A} + \mathbf{R}$ are

$$\mathbf{S} = \begin{bmatrix} 0 & -3 & -3 \\ -1 & -2 & -3 \\ -3 & -1 & -2 \end{bmatrix}, \quad \mathbf{E} = \begin{bmatrix} 3 & -3 & -3 \\ 0 & 0 & -2 \\ -1 & 0 & 0 \end{bmatrix}.$$

Therefore, the exemplar for point 1,2,3 is 1,1,2, i.e. the exemplar for point 2 is point 1 while point 2 is the exemplar of point 3. Hence, $\delta_k(\mathbf{c}) = -\infty$ in the objective function. Note that, since $e_{21} = e_{22}$ and $e_{32} = e_{33}$, it would be equally good to let the exemplar of point 2 be point 1 or point 2, and exemplar of point 3 be point 2 or point 3. In fact, these equivalent good assignment configurations for $\mathbf{E}$ have different $\sum_{i=1}^{N} s_{ic_i}$. However, AP cannot identify the best configuration $\mathbf{c}$ among them with maximal $\sum_{i=1}^{N} s_{ic_i}$ and constraint $\delta_k(\mathbf{c}) \neq -\infty$.

Example 2 In this example, we randomly generates two 2D points and duplicate them to obtain four points. Using negative Euclidean distance, max-sum AP with 100 iterations and $c_i = \arg\max_k \{a_{ik} + r_{ik}\}$ (when there is more than one maxima, we prefer $k \neq i$ to be c_i), we obtain the exemplar of point 1,2,3,4 are 1,4,1,2 respectively, in cases such as when

$$\mathbf{S} = \begin{bmatrix} -0.0323 & -0.0646 & 0 & -0.0646 \\ -0.0646 & -0.0323 & -0.0646 & 0 \\ 0 & -0.0646 & -0.0323 & -0.0646 \\ -0.0646 & 0 & -0.0646 & -0.0323 \end{bmatrix}.$$

Example 3 In this example, the input has only 2 points. The similarities are -1 between different data points, and the preferences are -1.000001 for both data points. We fix 20 iterations of message passing since there are only 2 data points. For max-sum AP [8] with damping factor = 0.5, the exemplar of point 1 is point 2, and the exemplar of point 2 is point 1. The implementation in [8] adds noises to $\mathbf{S}$. However, the same result exemplar assignment is obtained either with or without noises.

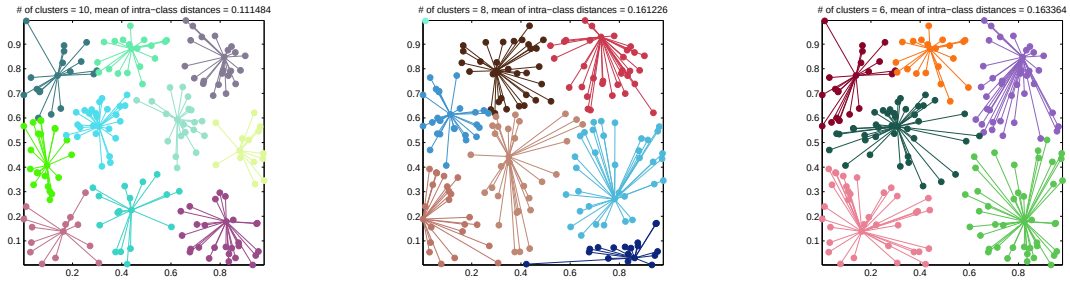
Discussions

Given the great similarity between Affinity Propagation (AP) and Expectation Maximization for fixed-parameter Mixture Model (MM-EM), we believe that AP and MM-EM may be two special cases for the same more general method, or AP can be regarded as a modified variation of MM-EM. As future work, it would be useful to systematically compare different update rules in Table 3.1.1 on various data sets by a third party, and consider a generalization beyond AP and MM-EM.

Table 3.1: **Various update rules.**

Method	Responsibility	Availability
Max-sum AP[111]	$r_{ik} = s_{ik} - \max_{j \neq k} \{a_{ij} + s_{ij}\}$ $r_{kk} = s_{kk}^{'} - \max_{j \neq k} \{a_{ij} + s_{ij}\}$	$a_{ik} = \min \left\{0, r_{kk} + \sum_{j=1, j \neq i, j \neq k}^{N} \max\{0, r_{jk}\}\right\}$ $a_{kk} = \sum_{j=1, j \neq k}^{N} \max\{0, r_{jk}\}$
Sum-product AP[110]	$r_{ik} = \frac{L_{ik}}{\sum_{j=1, j \neq k}^{N} a_{ij} L_{ij}}, r_{kk} = \frac{L_{kk}^{'}}{\sum_{j=1, j \neq k}^{N} a_{ij} L_{ij}}$	$a_{ik} = \frac{1}{\left(\frac{1}{r_{kk}} - 1\right) \frac{1}{\prod_{j=1, j \neq i, j \neq k}^{N} \left(1 + r_{jk}\right)} + 1}, a_{kk} = \prod_{j=1, j \neq k} (1 + r_{jk}) - 1$
Sum-prod Responsibility Max-sum Availability AP	$r_{ik} = \frac{L_{ik}}{\sum_{j=1, j \neq k}^{N} a_{ij} L_{ij}}, r_{kk} = \frac{L_{kk}^{'}}{\sum_{j=1, j \neq k}^{N} a_{ij} L_{ij}}$	$a_{ik} = \min \left\{0, r_{kk} + \sum_{j=1, j \neq i, j \neq k}^{N} \max\{0, r_{jk}\}\right\}$ $a_{kk} = \sum_{j=1, j \neq k}^{N} \max\{0, r_{jk}\}$
+n Max-sum AP	$r_{ik} = a_{ik} + s_{ik} - \max_{j \neq k} \{a_{ij} + s_{ij}\}$ $r_{kk} = a_{kk} + s_{kk}^{'} - \max_{j \neq k} \{a_{ij} + s_{ij}\}$	$a_{ik} = \min \left\{0, r_{kk} + \sum_{j=1, j \neq i, j \neq k}^{N} \max\{0, r_{jk}\}\right\}$ $a_{kk} = \sum_{j=1, j \neq k}^{N} \max\{0, r_{jk}\}$
+n Sum-product AP	$r_{ik} = \frac{a_{ik} L_{ik}}{\sum_{j=1, j \neq k}^{N} a_{ij} L_{ij}}, r_{kk} = \frac{a_{ik} L_{kk}^{'}}{\sum_{j=1, j \neq k}^{N} a_{ij} L_{ij}}$	$a_{ik} = \frac{1}{\left(\frac{1}{r_{kk}} - 1\right) \frac{1}{\prod_{j=1, j \neq i, j \neq k}^{N} \left(1 + r_{jk}\right)} + 1}, a_{kk} = \prod_{j=1, j \neq k} (1 + r_{jk}) - 1$
-e +n Max-sum AP	$r_{ik} = a_{ik} + s_{ik} - \max_{j} \{a_{ij} + s_{ij}\}$ $r_{kk} = a_{kk} + s_{kk}^{'} - \max_{j} \{a_{ij} + s_{ij}\}$	$a_{ik} = \min \left\{0, r_{kk} + \sum_{j=1, j \neq k}^{N} \max\{0, r_{jk}\}\right\}$ $a_{kk} = \sum_{j=1}^{N} \max\{0, r_{jk}\}$
+n -t Max-sum AP	$r_{ik} = a_{ik} + s_{ik} - \max_{j \neq k} \{a_{ij} + s_{ij}\}$ $r_{kk} = a_{ik} + s_{kk}^{'} - \max_{j \neq k} \{a_{ij} + s_{ij}\}$	$a_{ik} = \sum_{j=1, j \neq i}^{N} r_{jk}$ $a_{kk} = \sum_{j=1, j \neq k}^{N} r_{jk}$
-i Max-sum AP	$r_{ik} = s_{ik} - \max_{j \neq k} \{a_{ij} + s_{ij}\}$	$a_{ik} = \min \left\{0, r_{kk} + \sum_{j=1, j \neq i, j \neq k}^{N} \max\{0, r_{jk}\}\right\}$ $a_{kk} = \sum_{j=1, j \neq k}^{N} \max\{0, r_{jk}\}$
-e -t -i +n Sum-prod Responsibility Max-sum Availability AP	$r_{ik} = \frac{a_{ik} L_{ik}}{\sum_{j=1}^{N} a_{ij} L_{ij}}$	$a_{ik} = \frac{1}{N} \sum_{j=1}^{N} r_{jk}$ or $a_{ik} = \sum_{j=1}^{N} r_{jk}$
-m -c GMM-EM	$\widetilde{r}_{ik} = \frac{\pi_k L_{ik}}{\sum_{j=1}^{N} \pi_j L_{ij}}$	$\pi_k = \frac{1}{N} \sum_{j=1}^{N} \widetilde{r}_{jk}$ or $\pi_k = \sum_{j=1}^{N} \widetilde{r}_{jk}$
-m -c +i GMM-EM	$\widetilde{r}_{ik} = \frac{\pi_k L_{ik}}{\sum_{j=1}^{N} a_{ij} L_{ij}}, \widetilde{r}_{ik} = \frac{\pi_k L_{kk}^{'}}{\sum_{j=1}^{N} a_{ij} L_{ij}}$	$\pi_k = \frac{1}{N} \sum_{j=1}^{N} \widetilde{r}_{jk}$ or $\pi_k = \sum_{j=1}^{N} \widetilde{r}_{jk}$
-m +c GMM-EM	$\widetilde{r}_{ik} = \frac{\pi_k L_{ik}}{\sum_{j=1}^{N} \pi_j L_{ij}}$	$\pi_k = \frac{1}{N} \sum_{j=1}^{N} \widetilde{r}_{jk}, \quad \Sigma_k = \frac{\sum_{i=1}^{N} \widetilde{r}_{ik} (\mathbf{x}_i - \mathbf{x}_k)(\mathbf{x}_i - \mathbf{x}_k)^T}{\sum_{i=1}^{N} \widetilde{r}_{ik}}$
+m +c GMM-EM	$\widetilde{r}_{ik} = \frac{\pi_k L_{ik}}{\sum_{j=1}^{N} \pi_j L_{ij}}$	$\pi_k = \frac{1}{N} \sum_{j=1}^{N} \widetilde{r}_{jk}, \boldsymbol{\mu}_k = \frac{\sum_{i=1}^{N} \widetilde{r}_{ik} \mathbf{x}_i}{\sum_{i=1}^{N} \widetilde{r}_{ik}}, \Sigma_k = \frac{\sum_{i=1}^{N} \widetilde{r}_{ik} (\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T}{\sum_{i=1}^{N} \widetilde{r}_{ik}}$
+c Max-sum AP	$r_{ik} = s_{ik} - \max_{j \neq k} \{a_{ij} + s_{ij}\}$ $r_{kk} = s_{kk}^{'} - \max_{j \neq k} \{a_{ij} + s_{ij}\}$	$a_{ik} = \min \left\{0, r_{kk} + \sum_{j=1, j \neq i, j \neq k}^{N} \max\{0, r_{jk}\}\right\}$ $a_{kk} = \sum_{j=1, j \neq k}^{N} \max\{0, r_{jk}\}, \Sigma_k = \frac{\sum_{i=1}^{N} r_{ik} (\mathbf{x}_i - \mathbf{x}_k)(\mathbf{x}_i - \mathbf{x}_k)^T}{\sum_{i=1}^{N} r_{ik}}$
-m +c +i GMM-EM	$\widetilde{r}_{ik} = \frac{\pi_k L_{ik}}{\sum_{j=1}^{N} a_{ij} L_{ij}}, \widetilde{r}_{ik} = \frac{\pi_k L_{kk}^{'}}{\sum_{j=1}^{N} a_{ij} L_{ij}}$	$\pi_k = \frac{1}{N} \sum_{j=1}^{N} \widetilde{r}_{jk}, \Sigma_k = \frac{\sum_{i=1}^{N} \widetilde{r}_{ik} (\mathbf{x}_i - \mathbf{x}_k)(\mathbf{x}_i - \mathbf{x}_k)^T}{\sum_{i=1}^{N} \widetilde{r}_{ik}}$

+n = normalized, -n = unnormalized; +i = indented, -i = non-indented; +t = truncated, -t = non-truncated; +e = exclusive, -e = non-exclusive; +m = variable mean, -m = fixed mean; +c = variable covariance, -c = fixed covariance.



(a) +e -n +t max-sum AP (b) -e +n +t max-sum AP (c) +e +n -t max-sum AP

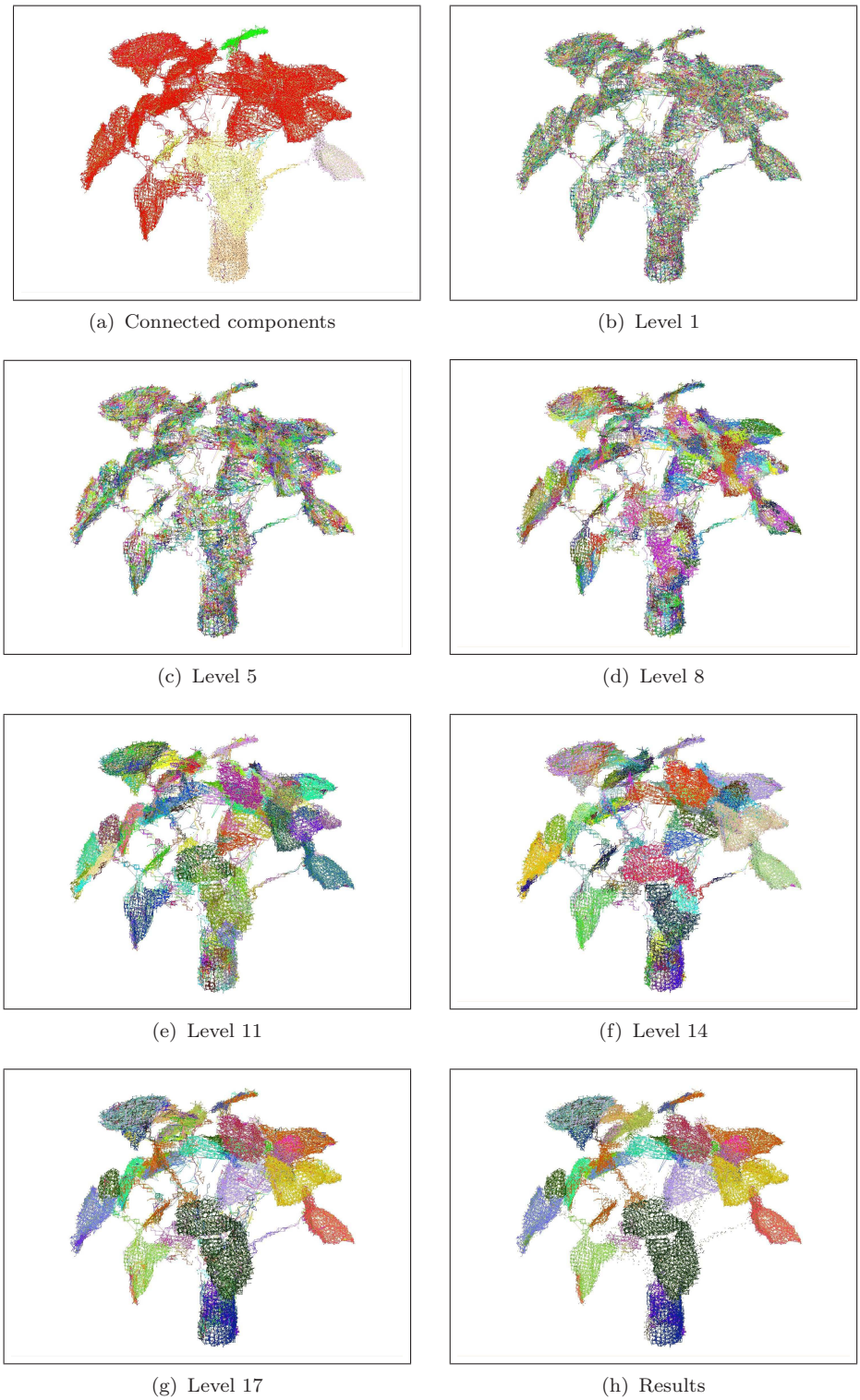
Figure 3.4: **Exclusion and truncation.** The corresponding update rules are shown in Table 3.1.1.

3.1.2 Hierarchical sparse affinity propagation

Affinity propagation on a sparse graph, called sparse affinity propagation, is more efficient as pointed in [111]. The implementation is similar to the description in Subsection 3.1.1 except that the responsibilities and availabilities are only updated on the connected edges. Then sparse affinity propagation runs in $O(T|\mathcal{E}|)$ time with T the number of the iterations and $|\mathcal{E}|$ the number of the edges. In our sparse graph, the time complexity is $O(Tn)$ since $|\mathcal{E}| = O(n)$.

We observed, however, according to the original sparse implementation in [111], the number of the data points that have the same exemplar i is at most $degree(i)$, where $degree(i)$ is the number of nodes connecting i. This is because point i, the exemplar for point k, must directly connect point k according to Equation 3.7 and the number of the points that connect point i is $degree(i)$. This will result in unexpectedly too many fragments as shown Figure 3.5(b).

To handle this problem, we propose a hierarchical sparse affinity propagation method. After obtaining the exemplars on the original sparse graph, we run again sparse affinity propagation on the exemplars by constructing a sparse graph on the exemplars and connecting the exemplars c_i and c_j if the point with its exemplar c_i is connected with at least one point whose exemplar is c_j. Then we can rerun sparse affinity propagation on the new exemplars until obtaining satisfactory results. Compared with the spectral clustering approach in [119], the hierarchical sparse affinity propagation is more efficient, running in $O(TLn)$ with T the number of the iterations and L the number of the hierarchies, and more effective. One example is shown in Figure 3.5.



(a) Connected components (b) Level 1

(c) Level 5 (d) Level 8

(e) Level 11 (f) Level 14

(g) Level 17 (h) Results

Figure 3.5: **Demonstration of hierarchical sparse affinity propagation.** (a) shows the connected components on the initial k-NN graph. (b)-(g) show the result of hierarchical sparse affinity propagation at different levels of iterations. (h) shows the final result.

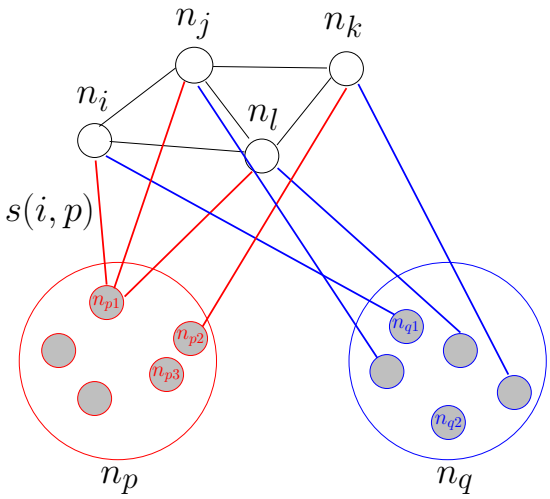


Figure 3.6: **Illustration of semi-supervised contraction.** n_p and n_q are candidate exemplars. The data points, n_i, n_j, n_l, n_k, are internally connected if they are neighbors, and are directly connected with the two candidate exemplars. n_p or n_q attracts competitively the data points in the semi-supervised AP algorithm.

3.1.3 Semi-supervised contraction

The original affinity propagation is an unsupervised clustering method. To utilize the partially labeled nodes, an efficient and effective semi-supervised affinity propagation method is proposed.

First, we group the nodes $n_{p1}, n_{p2}, \cdots$ that have the same known label, and contract these nodes into a single new node n_p. Similarly, $n_{q1}, n_{q2}, \cdots$ are contracted into a single node n_q. A toy example is shown in Figure 3.1.3. And we set the preferences of the contracted nodes to zero, i.e. $s(p,p) = s(q,q) = 0$ (this is equivalent that the exponential similarity is 1.).

Then, we update the new edges $\bar{\mathcal{E}}$ based on the original edges $\mathcal{E}$ and consider the similarities between all the remaining nodes $n_i \notin n_p \cup n_q$ and the contracted nodes n_p, n_q. We connect $n_i \notin n_p \cup n_q$ and n_p, n_q, and cut all the edges between the nodes in n_p, n_q. We set the similarities on the connected edges $\bar{\mathcal{E}}$ as follows.

1. For the similarity on $(n_i, n_j) \in \bar{\mathcal{E}}$ if $n_i \notin n_p \cup n_q$, $n_j \notin n_p \cup n_q$ and $(n_i, n_j) \in \mathcal{E}$, we just copy the similarity from the original weighted graph.

2. Considering the similarity on $(n_i, n_t) \in \bar{\mathcal{E}}$ if $p_t \in n_t$ and at least one p_t such that $(n_i, p_t) \in \mathcal{E}$, $n_t \in \{n_p, n_q\}$, we set it as the largest similarity between node n_t and any node $p_t \in n_t$ as $s(i,t) = \max_{p_t \in n_t} s(i, p_t)$.

3. For edge $(n_i, n_t) \in \bar{\mathcal{E}}$ if there is no point $p_t \in n_t$ such that $(n_i, p_t) \in \mathcal{E}$), we use

the distance of the shortest path between n_i and n_t to estimate their similarity $s(i,t) = \max_{path_{i,t}} \sum_{(j,k)\in path_{i,t}} s(j,k)$.

Finally, when the algorithm converged, availabilities and responsibilities are combined to identify exemplars. For point i, its corresponding label is obtained as

$$k^* = \arg\max_{k\in\{p,q\}}\{a(i,k) + r(i,k)\}. \tag{3.9}$$

Here, we perform the exemplar assignment only from the labeled point set to obtain semi-supervised contraction, which is slightly different from Equation 3.7. One demonstration is shown in Figure 3.8. Note that the algorithm can be easily generalized to more than two exemplars.

The semi-supervised affinity propagation propagates the message on the sparse graph by setting only the labeled nodes as candidate exemplars. It can converge in $O(Tn)$ time. Related discrete algorithms, such as iterated conditional mode, graph cuts, belief propagation, tree-reweighted message passing can also solve this problem [120] in more time complexity, and other relaxation algorithms, such as label propagation [121], may not obtain good performance.

3.2 Joint 2D and 3D segmentation

Structure from motion takes an image sequence of a rigid (or static) object as the input and recovers the camera poses and a cloud of 3D points. After many years of continuous research, nowadays, the structure from motion algorithm, such as [122], can robustly and accurately recover hundreds of thousands of points and all camera poses.

The reconstructed 3D points, however, are unstructured in space, therefore are not yet sufficient for creating a geometric model of the underlying objects. To structure the available 3D points and registered 2D images, recent researches [9, 119] show that a joint segmentation of the reconstructed 3D points and the multiple 2D images is fundamental for the subsequent modeling applications. Obviously, the concept of object is subjective, and learning from the user assisted 2D image segmentation gives the object segmentation more useful information. Hence, we wish to segment 3D points and 2D images into groups, where each group represents a distinct object. This segmentation can be regarded as a post-processing step of structure from motion, which provides semantic organizations of the recovered 3D points. And this is very useful for the subsequent 3D modeling of the scene.

The segmentation can be performed individually on the 3D points or 2D images. The 3D points, without considering the 2D images, is a little like the range data. The

segmentation is usually based on local geometric characterizations, which is insufficient to obtain semantic segmentation. On the other hand, segmenting 2D natural images is a well-studied topic, and many successful methods, such as [123, 124], were proposed. It is a choice to individually segment each image of the multiple view sequence. For example, a pure 2D segmentation approach to reconstruct small-leaf trees is proposed in [125]. However, it definitely misses very rich 3D information or 2D correspondence information, and hence can not obtain satisfactory results on large objects with more complex color, texture and shape information.

Some methods were proposed to utilize the motion information for multiple image segmentation. The layered approaches originated from [126] usually do not directly adopt 3D reconstruction information. In [127, 128], motion estimation and segmentation on the extracted correspondences between frames are performed, and then layer assignment (i.e. pixel label) is obtained through propagating the labels of the corresponding pixels. In [129], the joint inference of motion estimation and labeling is solved using the Expectation Maximization algorithm.

The modern stereo matching framework, such as [130], is very similar to the layered approach, and it in essence discretizes the 3D space into a few layers. Bilayer segmentation of binocular stereo video in [131], a simplest layered representation, probabilistically fused the stereo cues and learned appearance model to separate the figure from the background.

3.2.1 Graph-based formulation

Let $I = \{I_i\}$ be the set of n images with $i = 1, \ldots, n$. Each image I_i is represented by a set of regions, i.e. $I_i = \{(\mathbf{u}_k, P_k)\}$ with k up to the number set by the visible projections of the quasi-dense points in this view, and $\mathbf{u}_k$ is the projection coordinate in 2D image space and P_k is the corresponding patch. It is assumed that all the images are fully calibrated with respect to a common coordinate frame. We define a *joint point* $\mathbf{x}$ to be a vector composed of the 3D coordinates (x, y, z) of a point in space and all its corresponding patches P_i in all images, i.e. $\mathbf{x} = ((x, y, z), (\mathbf{u}_1, P_1), \cdots, (\mathbf{u}_n, P_n))$, where each projection satisfies $\mathbf{u}_i = \mathbf{P}_i(x, y, z, 1)^T$ for the projection matrix $\mathbf{P}_i$ of the i-th camera. The correspondence information is encoded in the joint point representation. And each joint point $\mathbf{x}$ is associated with an n-dimensional visibility vector $\mathbf{v}$ with binary values to indicate that $\mathbf{u}_i$ is visible in the i-th image if the i-th component is 1, and invisible otherwise. A segmentation is a set of labels $L = \{l_k\}$, and each of them l_k assigns a set of joint points to a common group. Here $X = \{\mathbf{x}_j\}$ is the given set of joint points, $V = \{\mathbf{v}_j\}$ is the given set of visibilities, and X and V are given by the quasi-dense reconstruction in our case. We now want to get the inference of L, given X, V and I. Similar to [119],

we define a weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with joint points as nodes, in which edge weights denote a local similarity measure between the two joint points in the graph $\mathcal{G}$. Different from [119], we generalize the joint point from a pixel level to a region (superpixel) level to help the definition of the joint similarities.

The set of edges $\mathcal{E}$ is constructed using the k-Nearest Neighbor (k-NN) technique. To guarantee that the joint points i and j, $(i, j) \in E$, must be both visible at least in one view, each view is associated with a set of joint points that are visible in this view. We then build for each view a k-NN network on the corresponding set of joint points according to the 3D Euclidean distance. Finally, we combine those networks together to reach a graph on the entire joint points.

3.2.2 Joint similarity

The joint use of 3D and 2D information for better segmentation, since all our images and 3D data are perfectly registered, is discovered by [9, 119]. All these useful information is encoded in the weights on the edge. For similar nodes, similar labels should be selected for them. Therefore, a similarity is defined on each edge to characterize the smoothness of the labels. The quality of segmentation fundamentally depends on the similarity, and hence we seek to define it jointly from both 3D and 2D features.

3D similarity The points that are closer in space tend to have a higher probability belonging to the same group, i.e. the distance between the points of the same group is smaller than that of the points in different groups. We naturally take this spatial distance as a similarity measure $s_{3d}(i, j) = -\frac{||\mathbf{p}_i - \mathbf{p}_j||^2}{2\sigma_{3d}^2}$, where σ_{3d}^2 is the expectation $E(||\mathbf{p}_i - \mathbf{p}_j||^2)$ and $\mathbf{p} = [x\ y\ z]^T$. In addition to the 3D Euclidean distance, the normal directions are also important for shape smoothness. We incorporate the difference between normal directions into the similarity and define $s_{3n}(i, j) = -\frac{||\mathbf{n}_i - \mathbf{n}_j||^2}{2\sigma_{3n}^2}$, where $\mathbf{n}_j$ is the normal direction vector of point j, approximately estimated from its neighbor points, and σ_{3n}^2 is the expectation $E(||\mathbf{n}_i - \mathbf{n}_j||^2)$. The final 3D similarity is given by $s_3(i, j) = s_{3d}(i, j) + s_{3n}(i, j)$.

2D color similarity Since a joint point $\mathbf{x}$ is associated with the image colors, we can define a similarity function encoding the color differences as $s_c(i, j) = -\frac{||E(\mathbf{c}_i) - E(\mathbf{c}_j)||^2}{2\sigma_c^2}$, where $\sigma_c^2 = E(||E(\mathbf{c}_i) - E(\mathbf{c}_j)||^2)$, and $E(\mathbf{c}) = \frac{1}{|\mathbf{v}|_1} \sum_{i=1}^{n} \mathbf{c}_i$. This color consistency between joint points is intuitively estimated using their average colors, since different points may have different numbers of visible color features. Averaging the colors leads to a more

stable solution. However, this similarity function only makes sense between the objects with apparent different colors.

In case of apparent similar colors, image contour features, similar to [132], should be incorporated into the similarity. It is assumed at present that each pixel $\mathbf{u}$ in view I_v is associated with a response $g_v(\mathbf{u})$ to show the degree of the pixel lying on a contour point. The endpoints of the edge (i, j) must both be visible at least in one view, meaning that the line segment $[i, j]$ must correspond to a line segment visible in the same view. We can use the following similarity measurement

$$s_{ic}(i,j) = -\frac{\text{med}_v\{\max_{t_v \in [i,j]_v} g_v(t_v)\}}{2\sigma_{ic}^2},$$

where the inner term $\max_{t_v \in [i,j]_v} g_v(t_v)$ finds the maximum contour response along the projected line segment $[i, j]_v$ in view v, the outer term $\text{med}_v\{\cdot\}$ tries to seek the median contour response in all possible views, and σ_{ic} is the variance of the median contour responses of all line segments. The response $g_v(\mathbf{u})$ is calculated from an edge map obtained by the similar orientation filter bank used in [132, 119].

Patch histogram similarity To utilize texture similarity, we express each patch vector in term of multi-resolution histograms [133]. That is, for each joint point, we collect all its patches $\{P_1, \ldots, P_k\}$ remained after filtering, and then build an average color histogram h_0. Without losing the spatial information, we further downsample the patches $t-1$ times and compute several normalized color histograms $h_1, \ldots, h_{t-1}$. Hence, a joint point now corresponds to a vector of histograms $h = [h_0, h_1, \ldots, h_{t-1}]$.

In this way, for any two joint points i and j, with the histogram representations h^i and h^j, their patch similarity is defined as

$$s_t(i,j) = -d(h^i, h^j) = -\frac{1}{t}\sum\nolimits_{k=0}^{t-1} d(h_k^i, h_k^j),$$

where $d(\cdot, \cdot)$ is the dissimilarity measures for histograms. Here, we choose the Kullback-Leibler divergence.

Finally, we are able to perform a simple addition of the similarities to define the joint similarity to be

$$s(i,j) = s_3(i,j) + s_c(i,j) + s_{ic}(i,j) + s_t(i,j).$$

3.2.3 Interactive segmentation

The concept of segmentation is obviously subjective. Hence, some user assistant information will greatly improve the segmentation. In recent years, interactive 2D image

(a) Strokes (b) Trimap (c) 3D projection

Figure 3.7: **User assistance.** (a) shows the strokes scribbled by the user. (b) shows the segmentation result in a trimap representation by our semi-supervised AP method. In (c), the 3D projections inside and outside the white-bounded region are assigned to different hard labels, and are used to propagate the labels into the other joint points invisible in this view using our semi-supervised AP method.

matting [134] is very successful, and a semantic segmentation was induced from a training example in [135]. Here, we use a similar way to allow the user conceptually group different objects in some 2D images. To specify an object, the user marks a few lines on the images by dragging the mouse cursor while pushing down a button. An example of our user-interface is shown in Figure 3.7(a), where different objects are marked by strokes with different colors.

We can segment 2D images using the semi-supervised contraction method, which is discussed in Subsection 3.1.3. We want to make use of these strokes and segmentation information to help segment the other views. This is more practical since we always have about 30 views and the user may not want to draw strokes on every view. Here, we make the assumption that all the surfaces in the scene are Lambertian. Under this assumption, the appearance models of all objects are roughly the same in all views. Hence, for the joint points with visible projections on this segmented 2D image, we directly set their labels the same as their projection's labels respectively, which can be obtained from segmented images. To handle the ambiguity of the projections near the boundary, such as the projections in the white area in Figure 3.7(c), we regard the joint points corresponding to them as unlabeled joint points.

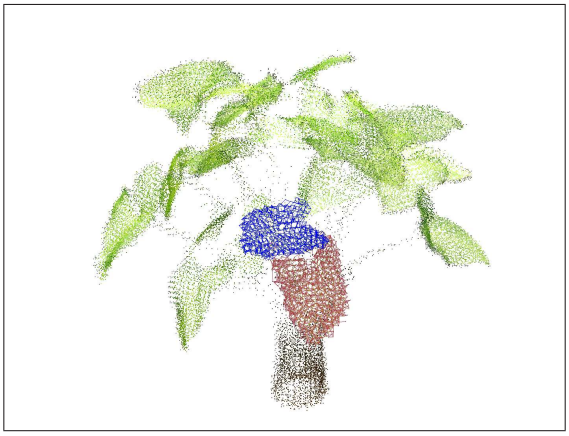
3.2.4 Experiments

Before performing our approach, we first run the connected component algorithm to extract connected components. One example is shown in Figure 3.5(a). In addition, before patch filtering, we run the bilateral filtering method to smooth all 2D images while keeping the edges.

(a) Selected group

(b) 2D projections

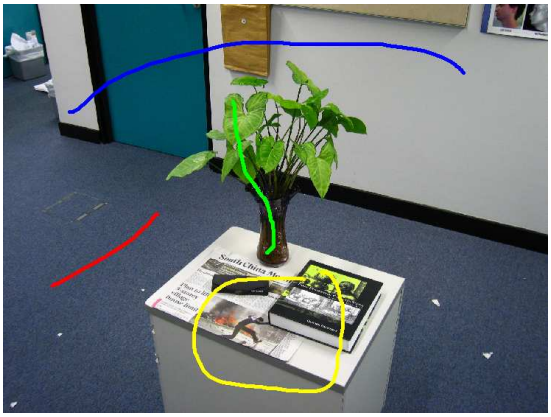
(c) Projection segmentation



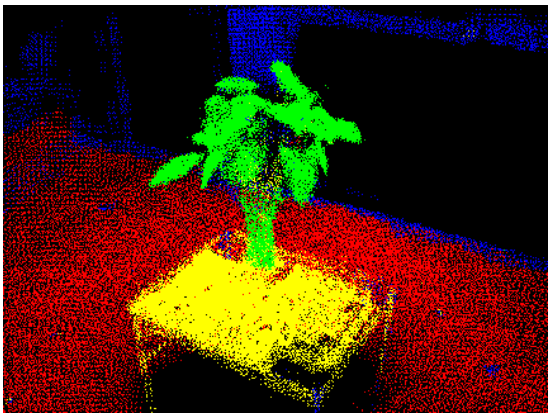
(d) 3D segmentation

Figure 3.8: **Demonstration of semi-supervised contraction.** (a) shows the selected cluster to be split. (b) shows the 2D projections visible on one view. (c) shows the separation of the visible 2D projections assisted by the user. (d) shows the clustering result on the selected group.

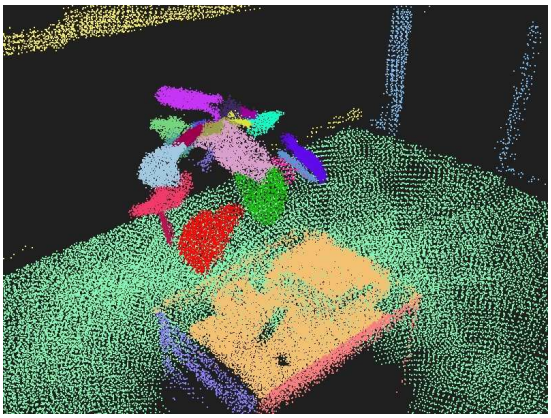
In Figure 3.9, we first draw four strokes in one view shown in Figure 3.9(a) to indicate that the scene consists of four major components: the tree, the desk, the ground and the wall. Then we learn the appearance models for each of the four components, respectively, and run the semi-supervised contraction on the 2D images to obtain 2D coarse segmentation and 3D segmentation as shown in Figure 3.9(b) by checking their projections. Thirdly, we run our hierarchical sparse affinity propagation and semi-supervised affinity propagation to obtain the grouping results on each component, and the final 3D segmentation and 2D segmentation results are shown in Figures 3.9(c) and 3.9(d) respectively. And in Figure 3.2.4 shows another example using the similar process. In all our experiments, without code optimization, the hierarchical sparse affinity propagation takes one to three minutes at lower levels and becomes real-time at higher levels, while the semi-supervised affinity propagation always provides results on the fly, which is suitable for user interaction.



(a) Strokes



(b) 3D coarse segmentation



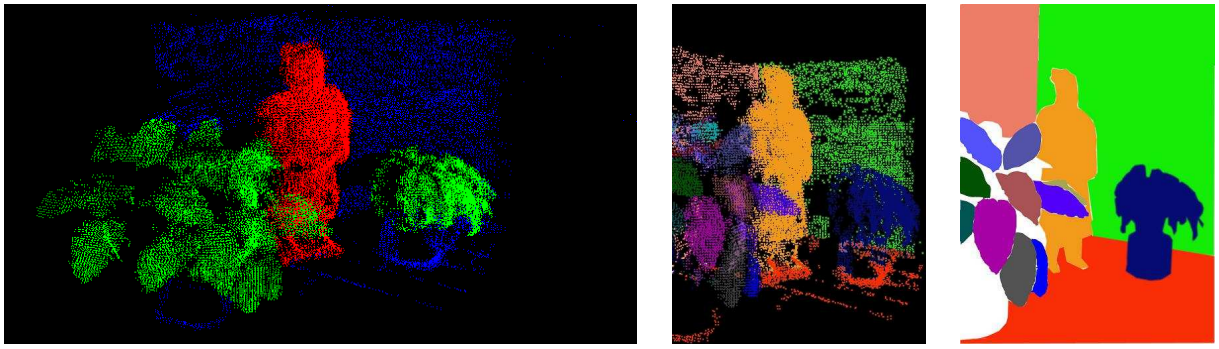
(c) Final 3D segmentation

(d) Final 2D segmentation

Figure 3.9: **Segmentation results for the office scene.** (a) shows the user assistance in one view to indicate the four components of the scene, and the 3D segmentation result is shown in (b) by utilizing the user assistance and our semi-supervised contraction algorithm. (c) shows the final 3D segmentation result using our hierarchical sparse and semi-supervised affinity propagation. (d) shows the corresponding 2D segmentation result.

As an application mentioned above, 3D modeling can benefit from satisfactory multiple view segmentation. A modeling example is shown in Figure 3.11. After we perform the proposed approach to obtain both 3D and 2D segmentation as shown in Figures 3.11(a) and 3.11(c), we can build the 3D surface and appearance models using the similar technique in [9]. The rendering result is shown in Figure 3.11(d).

Given both 2D images and 3D points reconstructed from those images, we proposed a joint segmentation approach to simultaneously segment 2D images and cluster 3D points. Efficient and effective hierarchical sparse and semi-supervised affinity propagation algorithms make the joint segmentation more practical. The results have demonstrated the powerfulness. Future work includes further study of the affects of different affinities.

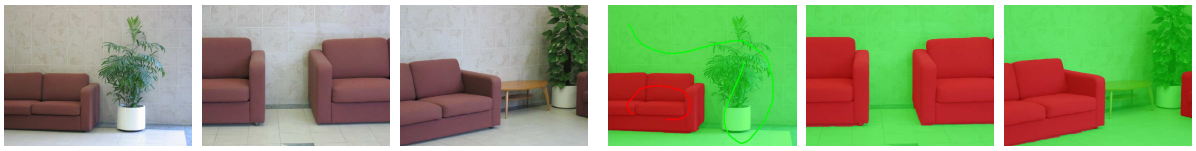


(a) Coarse 3D segmentation (b) Final 3D segmentation (c) Final 2D segmentation

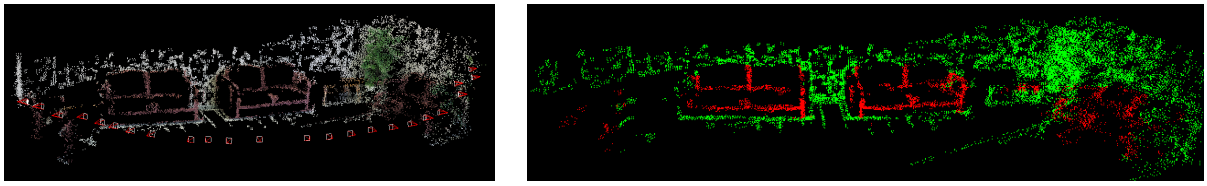
Figure 3.10: **Segmentation results for the terra-cotta warriors scene.** (a) shows the initial result using user assistance. (b) shows the final 3D segmentation result using our hierarchical sparse and semi-supervised affinity propagation. (c) shows the corresponding 2D segmentation result.

(a) 3D groups (b) Group projection (c) 2D leaves segmentation (d) Rendering result

Figure 3.11: **Segmentation results for the Nephthytis scene.** To be clear, this example only shows the segmentation results on the leaves. (a) shows the grouping result on 3D space, (b) shows the projections of the groups, (c) shows the image segmentation result, and in addition (d) shows the 3D modeling example by using the techniques in [9] based on our multiple view segmentation result.

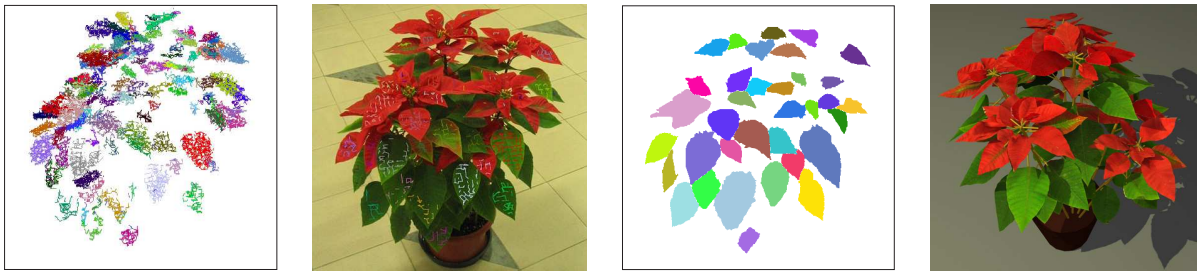


(a) 3 of the 24 Input Images (b) 2D Segmentation Result



(c) Reconstructed 3D Point Cloud with Camera Matrix (d) 3D Segmentation Result

Figure 3.12: **Segmentation results for the sofa scene.**



(a) 3D groups (b) Group projection (c) 2D leaves segmentation (d) Rendering result

Figure 3.13: **Segmentation results for the Poinsettia scene.** To be clear, this example only shows the segmentation result on the leaves. (a) shows the grouping result on 3D space, (b) shows the projections of the groups, (c) shows the images segmentation results, and in addition (d) shows the 3D modeling application of multiple view segmentation, the rendering result.

3.3 Semantic segmentation

3.3.1 Introduction

Understanding the semantic content of images is a fundamental and challenging problem in computer vision. In this work [136], we are especially interested in simultaneously learning object class models and performing segmentation on multi-view images captured along streets in outdoor environments. This problem has many potential applications, such as to automatic vehicles in the DARPA Urban Challenge, extensions to earth maps, and city modeling [64, 137].

Related works

In recent years, many methods have been proposed for simultaneous single-view multi-class object classification and segmentation, such as [138, 139, 140, 141, 142, 143, 144, 145]. In our setting, we have multi-view images of the same scene to improve the performance.

For object recognition tasks, several multi-view systems have been proposed such as [146, 147, 148, 4, 149]. In these methods, either multi-view or 3D information is utilized during training. However, all of these methods focus on single view recognition during testing, while our problem is to recognize and segment multiple views during both training and testing.

[150] proposed a system making use of multi-view information during testing for instance-level retrieval. However, they focus on distributed systems in which the computation power and transmission bandwidth are limited. [151] proposed a joint affinity propagation method for both automatic segmentation and interactive refinement. Al-

though multi-view information is used, this clustering-based approach cannot infer semantic meaning for object classes.

The most related work is probably [152], where Brostow *et al.* proposed to utilize 3D information for street image segmentation. Compared with [152], our approach differs in the following aspects. First, we propose a graph-based optimization approach to enforce consistency of the segmentation result across multiple views. Second, we adaptively select the training data from a large label pool. Furthermore, we propose a simple yet powerful approach for data labeling tasks, while [152] releases a high-quality labeled data set. Our approach works on practical data collected by Google Street View without human intervention. The data are very noisy and have strong glare. Posner *et al.* [153] also worked on similar problems although using range data.

Overview

We propose a multi-view semantic segmentation framework for images captured by a camera mounted on a car driving along the street. In Section 3.3.2, we illustrate how to set up the image capturing system. The pixel correspondences are then obtained across multiple views. Structure from Motion is used to reconstruct the scene geometry and prune incorrect correspondences. With both 2D and 3D information available, in Section 3.3.3, we lay out a Markov Random Field (MRF) across multiple images. Nodes in MRF represent superpixels from images, while edges represent smoothness across either neighboring superpixels in the same images or from different images linked by pixel correspondences. Section 3.3.3 gives the definition of the unary data term, while Section 3.3.3 defines the smoothness term. To improve performance by scene alignment, Section 3.3.3 illustrates the organization of the label pool to ease similar context and learning transference. In Section 3.3.4, we propose a approach to enable labeling of many images at the same time using the available geometry and color information. Finally, we demonstrate our approach in Section 3.3.5.

3.3.2 Preprocessing

We use a camera that usually faces the building façade and moves laterally along streets. The camera should be preferably held straight and the neighboring two views should have sufficient overlapping. The top view of the camera motion is illustrated in Figure 3.14. With the captured images, we first compute pixel-to-pixel correspondences between two adjacent images using a robust uncalibrated matching algorithm [154, 155]. Taking an image in sequence as a bridge, we can obtain feature tracks of three neighboring views

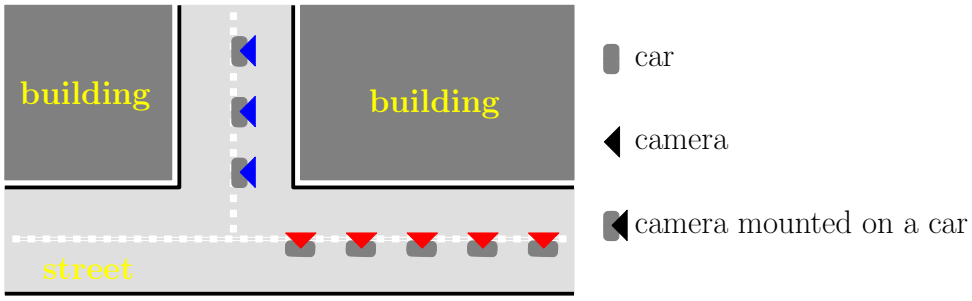


Figure 3.14: **Top view of the camera motion.** The car drives along the street and makes a 90-degree turn at the corner. Therefore, we break the sequence down into two different sequences, denoted in red and blue at the turn.

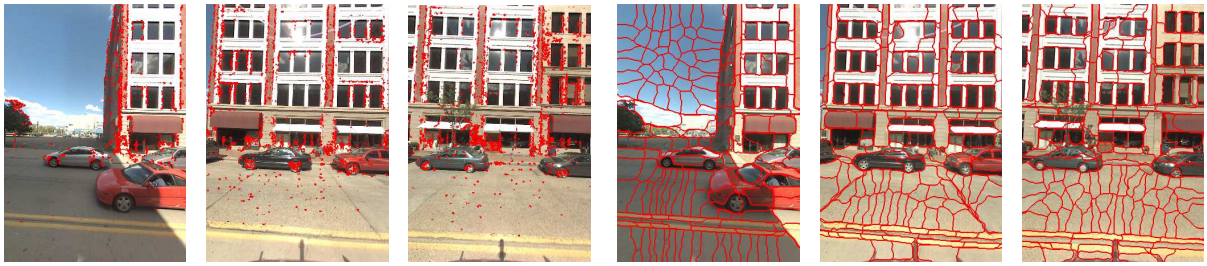
for projective reconstruction. We merge all the triplets by estimating the transformation between those with two common images [151, 4] and metrically upgrade to Euclidian space. In each step, bundle adjustment is used to minimize the geometric errors, and feature tracks are merged and linked to cover more views.

Figure 3.3.2 shows an example of feature tracks across multiple view images and corresponding 3D reconstruction. We not only recover a set of 3D points representing the scene, but also all camera poses and parameters. We denote a feature track as $\mathbf{t} = \langle \mathbf{x}, (x_i, y_i, i), (x_j, y_j, j), \dots \rangle$, where $\mathbf{x} = (x, y, z)$ is the coordinate for the corresponding 3D point, and (x_i, y_i, i) is the 2D projection (x_i, y_i) on the i-th image, I_i.

We work on sequences with about 100 images and break down at the turn in the driving path as exemplified in Figure 3.14. To ease the description of the 3D geometry, the right-hand coordinate system is rotated to align with the average down vector of all reconstructed cameras to be in the y direction, and the camera path is roughly on the x axis, while the orientations of the cameras are roughly the same as the $+z$ direction. To improve the segmentation accuracy and speed up the process, we over-segment [156, 157] each input image, I_i, into about 200 superpixels $\{p_j\}$.

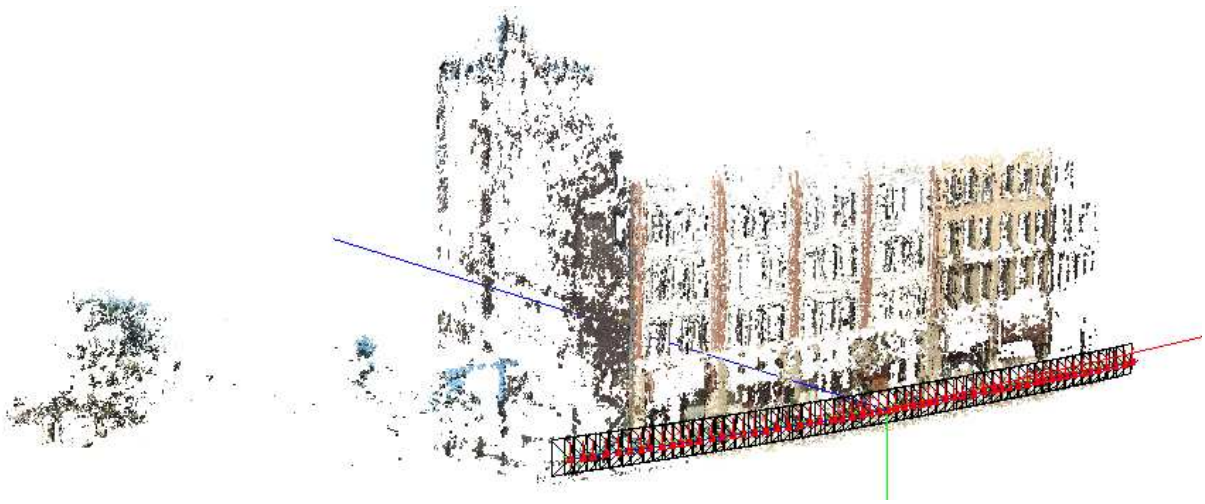
3.3.3 Multi-view semantic segmentation

Since street view data usually contain multiple images, we define a Markov Random Field for the entire sequence to improve the segmentation consistency across different views. For each image, I_i, we build a graph, $\mathcal{G}_i = \langle \mathcal{V}_i, \mathcal{E}_i \rangle$, on the over-segmentation results. Each vertex, $p \in \mathcal{V}_i$, in the graph is one superpixel in the over-segmentation, while the edges, $\mathcal{E}_i$, denote the neighboring relationship between superpixels. The graphs $\{\mathcal{G}_i\}$ from multiple images in the same sequence are merged into a large graph, $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, by adding the edges between two superpixels in correspondence across different views. The superpixels,



(a) Tracks across multiple views.

(b) Superpixel segmentation.



(c) 3D reconstruction by Structure from Motion. The x, y and z axes are indicated in red, green and blue, respectively.

Figure 3.15: **Preprocessing.**

p_i and p_j, from images I_i and I_j are in correspondence if and only if there is at least one feature track, $\mathbf{t} = \langle \mathbf{x}, (x_i, y_i, i), (x_j, y_j, j), \dots \rangle$, with projection (x_i, y_i) lying inside superpixel p_i in image I_i, and projection (x_j, y_j) lying inside superpixel p_j in image I_j. To limit the graph size, there is at most only one edge, e_{ij}, between any superpixel, p_i and p_j, in the final graph, $\mathcal{G}$.

The labeling problem is to assign a unique label, l_i, to each node, $p_i \in \mathcal{V}$. The solution, $L = \{l_i\}$, can be obtained by minimizing a Gibbs energy [158]

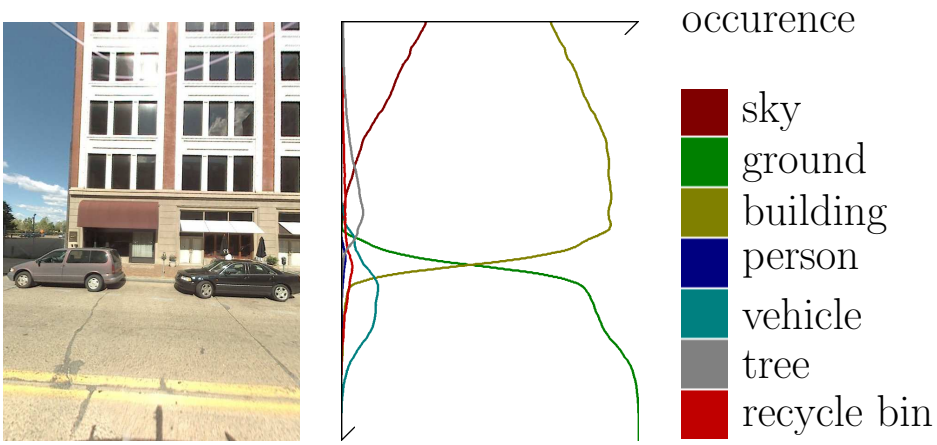
$$E(L) = \sum_{p_i \in \mathcal{V}} \psi_i(l_i) + \rho \sum_{e_{ij} \in \mathcal{E}} \psi_{ij}(l_i, l_j). \tag{3.10}$$

Since the smoothness costs defined in Section 3.3.3 satisfy the metric requirement, after the cost has been computed, GraphCut-based alpha expansion [159] can be used to obtain a local optimized label configuration, L.

Unary potential

To define the unary potential function, $\psi_i(\cdot)$, we extract features for superpixels to train discriminative classifiers.

2D features For each superpixel, p_i, we compute a 192-dimensional feature description vector, $\mathbf{f}_i^A$, based on the 2D image-based appearance. Built on [160, 161, 162, 141], for each superpixel, p_i, the feature vector, $\mathbf{f}_i^A$, contains the median, deviation, skewness and kurtosis statistics over the superpixel, p_i, of the RGB and Lab color-space components, as well as the texture features drawn from filter bank responses. The filter bank we used is made of three Gaussians, four Laplacians of Gaussians and four first-order derivatives of Gaussians. This filter bank has been shown to achieve good performance in [162] among a number of different filter combinations of derivatives of Gaussians and Gabor kernels. In addition, following [160], we compute the size and shape of each superpixel. The shape features consist of the ratio of the region area to the perimeter square, the moment of inertia about the center of the mass, and the ratio of the area to the bounding rectangle area. As in [161], we also append to the description vector the average of the descriptor over the neighbors for each superpixel weighted by the number of pixels for the neighbors.



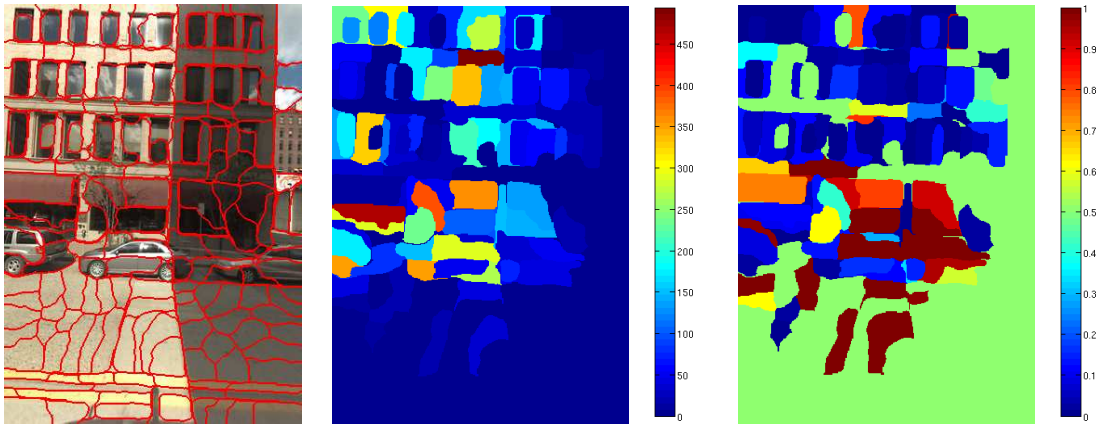
(a) An example image (b) Distribution

Figure 3.16: **Pixel location statistics.**

Because of the way we captured images, we can roughly learn the location for each class of objects. For example, the sky is always in the upper part of the image, while the ground is always in the lower part. Since our camera moves laterally along the street, each pixel position at the same height in the image space should have the same chance to be a specific class. To illustrate the idea, we compute the accumulated frequency of different classes from all labeled data and plot the distribution in Figure 3.16. Based on this observation, we only use the vertical position of the superpixel as the one-dimensional feature vector, $\mathbf{f}_i^P$.

3D features We define the superpixel orientation and the 3D point density as our geometric features, $\mathbf{f}_i^G$. We do not use the absolute height above the camera and the absolute distance to the camera path as in [152], because an extra setup for the capturing

system to measure the absolute dimensions is needed. We also do not use the back projection residual since it strongly depends on the implementation of Structure from Motion.



(a) Superpixel (b) Density (c) Direction

Figure 3.17: **An example of 3D geometric features.**

Let $\mathcal{T}_i$ denote all tracks that have projection in p_i, and let $\mathbf{m}_i$ be the medians of three components of all 3D points in $\mathcal{T}_i$. For each superpixel, p_i, the patch normal, $\mathbf{n}_i$, is provided by the symmetric 3×3 positive semidefinite matrix, $\sum_{\mathbf{x} \in \mathcal{T}_i} (\mathbf{x} - \mathbf{m}_i) \otimes (\mathbf{x} - \mathbf{m}_i)$. Among the eigenvectors, $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$, associated with the eigenvalues, $\lambda_1 \geq \lambda_2 \geq \lambda_3$, respectively, we choose $\mathbf{n}_i$ to be either $\mathbf{v}_3$ or $-\mathbf{v}_3$. The sign is chosen to have a greater than 180-degree angle between $\mathbf{n}_i$ and the camera orientation. In the experiments, we only estimate the normal direction for regions containing at least five 3D points. The estimated normal direction, $\mathbf{n}_i$, is projected onto the yz-plane. The dot product of the normalized unit projected vector and the $-y$ direction is defined to be the orientation descriptor. For diluted regions without sufficient points for normal estimation, we let this feature value to be 0.5. This definition of the geometric features is very useful for classification between the ground with normals roughly the same as $-y$, and other objects such as buildings. For objects that are textureless, such as the sky, we use the density $|\mathcal{T}_i|$ of the feature tracks to distinguish them.

The boosting classifier The collection of all feature descriptors are then whitened to give a zero mean and unit covariance. We learn a series of one-vs-all AdaBoost classifiers [163] for each class label l. Here, we take as positive examples the superpixels that belong to that class in the ground-truth labeling and as negative examples all superpixels with ground-truth labels of different classes. We apply the AdaBoost classifier that we have learned for each class, l, to the descriptors. The estimated confidence value can be

reinterpreted as a probability distribution using softmax transformation:

$$P_i\left(l_i|\mathbf{f}_i^A,\mathbf{f}_i^P,\mathbf{f}_i^G\right)=\frac{\exp\left(H\left(l_i,\mathbf{f}_i^A,\mathbf{f}_i^P,\mathbf{f}_i^G\right)\right)}{\sum_l\exp\left(H\left(l,\mathbf{f}_i^A,\mathbf{f}_i^P,\mathbf{f}_i^G\right)\right)},\tag{3.11}$$

where $H\left(l,\mathbf{f}_i^A,\mathbf{f}_i^P,\mathbf{f}_i^G\right)$ is the output of the AdaBoost classifier for class l. We then define the unary potential as $\psi_i\left(l_i\right)=-\log P_i\left(l_i|\mathbf{f}_i^A,\mathbf{f}_i^P,\mathbf{f}_i^G\right)$.

Smoothness

For edge $e_{ij}\in\mathcal{E}_k$ in the same image, I_k, the smoothness cost is defined as

$$\psi_{ij}\left(l_i,l_j\right)=\left[l_i\neq l_j\right]\cdot g\left(i,j\right),\tag{3.12}$$

where

$$g\left(i,j\right)=\frac{1}{\zeta\left\|\mathbf{c}_i-\mathbf{c}_j\right\|^2+1}\tag{3.13}$$

and $\left\|\mathbf{c}_i-\mathbf{c}_j\right\|^2$ is the $L2$-Norm of the RGB color difference of two superpixels, p_i and p_j. Note that $[l_i\neq l_j]$ allows us to capture the gradient information only along the segmentation boundary. In other words, ψ_{ij} is a penalty term when adjacent nodes are assigned with different labels. The more similar the colors of the two nodes are, the larger ψ_{ij} is, and thus the less likely the edge is on the segmentation boundary.

For edge $e_{ij}\in\mathcal{E}$ across two images, the smoothness cost is defined as

$$\psi_{ij}\left(l_i,l_j\right)=\left[l_i\neq l_j\right]\cdot\lambda\left|\mathcal{T}_{ij}\right|g\left(i,j\right)\tag{3.14}$$

where $\mathcal{T}_{ij}=\{\mathbf{t}=\langle\mathbf{x},\left(x_i,y_i,i\right),\left(x_j,y_j,j\right),\dots\rangle\}$ is the set of all feature tracks with projection (x_i,y_i) lying inside the superpixel, p_i, in image I_i, and projection (x_j,y_j) lying inside the superpixel, p_j, in image I_j. This definition encourages two superpixels with more feature track connections to have the same semantic segmentation label, since the cost to have different labels is high due to large $|\mathcal{T}_{ij}|$.

Adaptive training

For each testing sequence, we only select a subset of labeled images that are similar with the input sequence as the training data for that sequence [164]. We define the distance between two images as the distance between their respective Gist descriptors [165]. The Gist descriptor is used because it has been shown to work well for retrieving semantically and structurally similar scenes. We create a Gist descriptor for each image

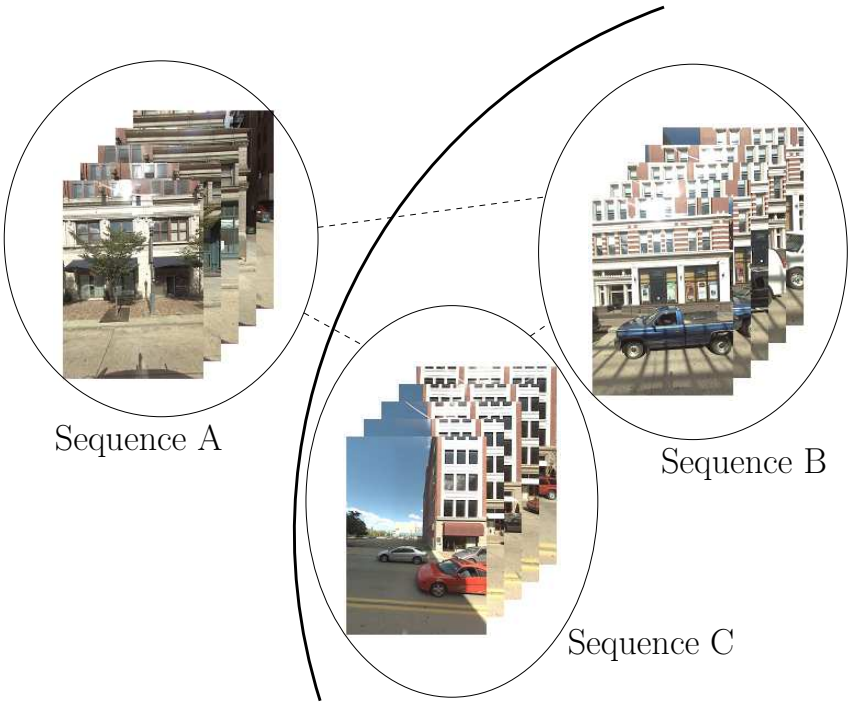


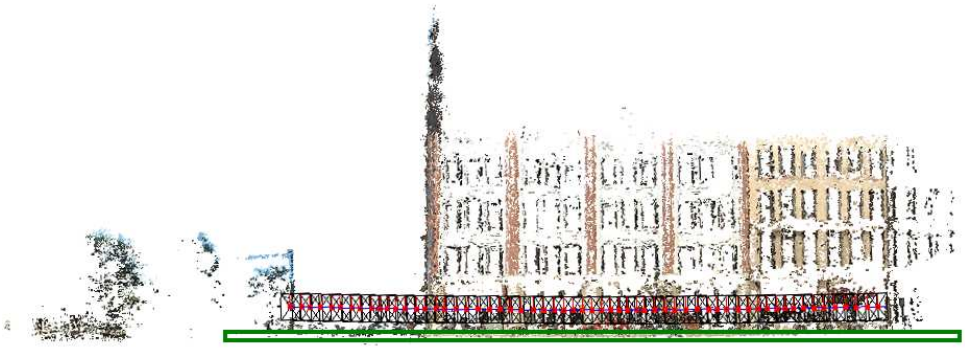
Figure 3.18: **Affinity clustering in the label pool.**

with 4 by 4 spatial resolution where each bin contains that image region's average response to Steerable filters at 4 scales with 8,8,4 and 4 orientations, respectively.

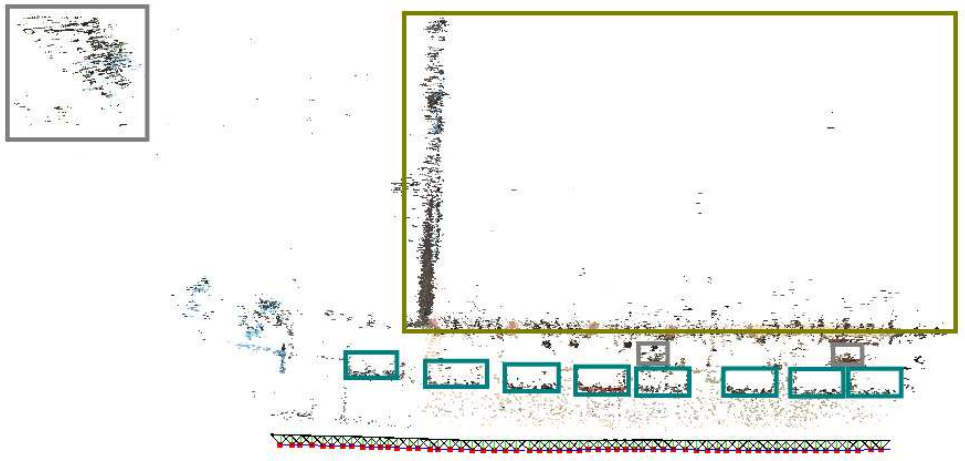
To speed up the training and prediction process, we cluster the labeled sequences in the pool based on affinity. As shown in Figure 3.18, we regard each label sequence as one node in a graph. The weight of the edge between each pair of sequences is defined to be the minimal Gist distance between any image in a sequence and any image in another sequence. Given this graph, we use Affinity Propagation [166] to cluster 40 labeled sequences into 7 clusters. We then learn 7 models respectively by first training the AdaBoost classifier in Section 3.3.3, and learning ρ in Equation 3.10, ζ in Equation 3.13, and λ in Equation 3.14 by piecewise training [141].

Given a testing sequence, we can compute all the distances between the labeled images and each image in the input sequence. We may define the distance between the testing sequence and one cluster as the minimal Gist distance between any image in the testing sequence and any image from the cluster. However, this process is very time consuming. Therefore, we approximate by using the middle image in each sequence. In this way, we only need to compute 40 distances between the Gist of the middle image in the testing sequence, and the pre-computed Gist descriptors for images at the middle positions of the 40 labeled sequences. We find the most similar cluster for the testing sequence and use the corresponding model for prediction.

3.3.4 Large-scale labeling



(a) Labeling at the front view.



(b) Labeling at the top view.

Figure 3.19: **Labeling in 3D.** Color code: ☐ ground, ☐ building, ☐ vehicle, ☐ tree.

Labeling a sufficient number of examples is a fundamental requirement for any supervised learning method. Many researchers in this field have proposed methods to ease labeling tasks, such as LabelMe [167], ESP game [168] and Peekaboom [169]. However, most of these methods tried to make the labeling become interesting and online to encourage people to do more labeling, while the efficiency has not yet been greatly addressed, possibly due to the single image nature of these tasks. In our multiple view setting, it is possible to enable large-scale labeling with little interaction. We first reconstruct the 3D scene for each sequence with about 100 images, and let the user label the 3D points in the 3D space. Using labels of 3D points, we are able to segment the 2D images at the same time. In this way, labeling once gives us about 100 labeled images. This significantly improves the efficiency.

In detail, after the 3D scene is reconstructed, as shown in Figure 3.19, the user can draw rectangular or polygonal regions to indicate the semantic meaning of the point

(a) Strokes (b) Segment regions (c) Info source likelihood

Figure 3.20: **Labeling in 2D.** Color code: ■ sky, ■ vehicle.

clouds. Note that the user may not want to, or may not be able to, identify and label all 3D points. The task is to use these non-perfect labeled 3D points to segment multiple view images. The same framework that we proposed previously can be naturally used in our labeling task. In Equation 3.10, the same smoothness defined in Section 3.3.3 can also be used since it involves no information gained from training data. The unary potential is defined as follows.

First of all, a superpixel, p_i, has a set, $\mathcal{T}_i$, of 2D projections of 3D points inside the superpixel region. The more points in $\mathcal{T}_i$ that are labeled, the more confidence we gain about the label of the region. Therefore, we define

$$P_i^{3D}(l_i = l) \propto \left|\mathcal{T}^l \cap \mathcal{T}_i\right| + \frac{\left|\mathcal{T}^{\text{unknown}} \cap \mathcal{T}_i\right|}{n}, \tag{3.15}$$

where $\mathcal{T}^l = \{\mathbf{t}|\mathbf{t}$ is a track labeled as class l by the user in 3D$\}$, $\mathcal{T}^{\text{unknown}}$ is the set of feature tracks that have no label information from the user, and n is the total number of all possible labels. This definition will put a uniform uncertainty on each class if there are unlabeled 3D points with projections in the superpixel region. However, it cannot characterize the projection density in each superpixel region. A superpixel region with more labeled projections should have more influence on the neighboring regions. The superpixel that has lower uncertainty should also contribute more. Therefore, we define the unary potential to be

$$\psi_i(l) = -\frac{\left|\mathcal{T}_i - \mathcal{T}^{\text{unknown}}\right| + \epsilon}{H(P_i(\cdot)) + \varepsilon} \log P_i(l), \tag{3.16}$$

where $\left|\mathcal{T}_i - \mathcal{T}^{\text{unknown}}\right|$ is the number of labeled feature tracks with projections in superpixel p_i, $H(P_i(\cdot))$ is the entropy of the distribution $P_i(\cdot)$, and ϵ and ε are two small positive values to avoid 0. $P_i(l)$ is set to be $P_i^{3D}(l)$.

Ideally, this method works well to identify regions with sufficient texture. However, for the classes lacking texture, such as the sky, where almost no 3D points are reconstructed, it is impossible to label them in 3D. Therefore, we also provide mechanism to draw strokes on one or more 2D images. When a superpixel in one image is covered by the strokes drawn by the user to be class l, the corresponding unary potential is set to $\psi_i(l_i = l) = -\infty$ and $\psi_i(l_i \neq l) = +\infty$. By adding these hard constraints, together with the definition of smoothness based on the color difference in Section 3.3.3, the labeling results can be obtained by MRF optimization.

In the same city block, the color distributions across instances of the same class are quite similar, while those across instances of different classes are always different. To draw as few strokes as possible, we want to make use of the strokes in one image, I_k, to segment the other images. To integrate this idea into our framework, for a superpixel, p_i, in an image, I_j, without strokes, we first distinguish whether the labeling information of p_i should come from 2D color or 3D points. Illustrated in Figure 3.20, we segment some regions in image I_k with the strokes and compute the color statistics of the regions, $\mathcal{R}$, belonging to the same set of classes for the strokes. The color distribution of all pixels in $\mathcal{R}$ is approximated by a Gaussian mixture model, $M_{\mathcal{R}}$, on the RGB color space. Furthermore, the color distribution of all pixels in $I_k - \mathcal{R}$ is approximated by another Gaussian mixture model $M_{I_k - \mathcal{R}}$. For a superpixel, p_i, in an image, I_j, without strokes, we determine the likelihood that the label information of p_i with mean color $\mathbf{c}_i$ comes from the 2D color from

$$P_i^{col}(\mathbf{c}_i) = \frac{P(\mathbf{c}_i|M_{\mathcal{R}})}{P(\mathbf{c}_i|M_{\mathcal{R}}) + P(\mathbf{c}_i|M_{I_k - \mathcal{R}})}. \tag{3.17}$$

The probability is defined accordingly by

$$P_i(l) = P_i^{col}(\mathbf{c}_i) P_i^{2D}(l) + \left(1 - P_i^{col}(\mathbf{c}_i)\right) P_i^{3D}(l), \tag{3.18}$$

where $P_i^{3D}(l)$ is defined in Equation 3.15, and $P_i^{2D}(l)$ is the color likelihood computed from Gaussian mixture model of pixel colors in the stroke-covered regions of image I_k that belong to class l. This definition is then used in Equation 3.16. If multiple views are labeled with 2D strokes, we just put all the pixels from the multiple views together to define $M_{\mathcal{R}}$ and $M_{I-\mathcal{R}}$ accordingly.

3.3.5 Experiments

	sky	ground	building	person	vehicle	tree	recycle bin
sky	**94.6**	-	5.1	-	-	-	0.1
ground	-	**97.6**	1.0	-	1.1	0.1	0.2
building	-	0.6	**98.6**	-	0.2	0.4	0.2
person	-	6.6	85.0	**8.3**	-	-	-
vehicle	-	15.5	3.8	1.6	**78.7**	0.2	0.2
tree	0.1	1.2	5.7	-	0.1	**92.8**	0.1
recycle bin	-	6.1	25.7	-	-	-	**68.1**

Table 3.2: **Accuracy of our approach to the evaluation data set in percentage.** This confusion matrix shows the pixel-wise recall accuracy for each class (rows) and is row-normalized to sum to 100%. Row labels indicate the true classes and column labels the predicted classes.

Figure 3.21: **Example Results.** Color code: sky, ground, building, person, vehicle, tree, recycle bin.

In our experiment, we use more than 10,000 images captured in the downtown of Pittsburgh by Google Street View. We want to segment the images into seven classes: sky, ground, building, person, vehicle, tree and recycle bin. All of our input images are at 640×905 resolution. We train our models with a small computer cluster composed of seven desktop PCs. Each sequence contains about 200,000 3D points. It takes about 2 days for training using all labeled sequences excluding over-segmentation. For testing, each image takes 25.7 seconds on average in one desktop PC excluding over-segmentation.

We first manually label every image in one sequence. Then, we label this sequence in the 3D space with a few strokes in 2D images and learn ρ in Equation 3.10, ζ in Equation 3.13, and λ in Equation 3.14 for large-scale labeling by piecewise training [141]. Now, we use the large-scale labeling method to randomly label another 39 sequences. Together with the previous one, we have totally 40 labeled sequences and 3,877 labeled images in the label pool.

For performance evaluation, we manually label 320 images sampled uniformly from the testing data set. The segmentation performance is measured as the global accuracy, i.e., the total proportion of pixels that are correct. The global pixel-wise prediction accuracy is 94.7% for the full model, 91.1% for the model from a random cluster (also below), 83.3% for the model without cross-view consistency, 81.2% for the model without smoothness, 75.4% for the model without 3D geometry features and smoothness, 69.9% for 2D appearance features together with the AdaBoost classifiers. Some example results predicted by the full model are shown in Figure 3.21. Note that there may be several images between two adjacent images not shown due to space limits. The confusion matrix of full model is presented in Table 3.2 for the pixel-wise recall accuracy per class.

We can see that the accuracy for the sky, ground, building and tree classes are very high. However, for the person class, the results are unsatisfactory for several reasons. First, objects like persons are difficult to reconstruct by Structure from Motion, since they are dynamically deforming and unclear due to small pixel coverage, or there is a large disparity due to the close proximity to the camera. There are very few 3D points for these objects and it is difficult for the user to label them in 3D. Hence, very few of our labeled data contains good labels of these objects. Second, these small objects are not suitable for our super-pixel representation. The traditional sliding windows approach can better encode the profile shape for such small objects.

This section tries to propose a consistent framework for multi-view segmentation. Although it is beyond the scope of this thesis, we can easily incorporate existing detection methods to handle these difficult classes. For example, in Figure 3.22, we use a state-of-the-art model [170] to obtain a bounding box of the person. With our segmentation

(a) Input Image (b) Our result (c) Detection [170] (d) Pruning

Figure 3.22: **An example of person detection.** Color code: ■ ground, ■ building, □ person, □ parts of body.

results for background objects, such as buildings and the ground, we can naturally put the boxes in perspective [171, 172] in order to prune incorrect predictions. For example, a person must stand on the ground and has a head usually at the pixel-position of the building. Finally, we can obtain the silhouette from the bounding box by methods such as GrabCut [173] or Active Contours.

In this section, we propose a multiple view framework for semantic object segmentation and demonstrate our approach on large-scale data sets from Google Street View images. Interesting future work will consider real-time implementation for prediction, better handling of small objects such as the illustration in Section 3.3.5, and extend the method to more general contexts beyond Street View.

CHAPTER 4

MODELING

Given the segmented 3D point cloud and 2D images, we are ready to produce the mesh model. The first thing that we need to decide is how to represent a façade of building. In Section 4.1, we propose a 2.5D representation for a façade, i.e., we represent a façade as a 2D depth map on the orthographic view. In Section 4.2, we discuss about how to analyze the structure of the façade. For example, how to detect the geometric elements embedded on the façade, such as windows. In Section 4.3, we discuss about how to compute a regular boundary for each façade. When the embedded geometric elements and boundaries are determined, the shape for the façade is fully recovered. In Section 4.4, the texture mapping methods of this shape is discussed. Finally, in Section 4.5, we provide some convenient way to interactively edit the result if needed.

4.1 Representation

The reconstructed 3D points are often noisy or missing due to varying textureness as well as matching and reconstruction errors. Therefore, we introduce a building regularization method in the orthographic view of the façade for structure analysis and modeling. Orthographic depth map and texture image are composed from multiple views in Section 4.1.1, and provide the working image space for later stages. In this way, we are able to transfer the reconstructed 3D information and segmentation information onto this 2D orthographic view. In Section 4.1.2, we discuss about the data structure of this 2.5D representation.

4.1.1 Inverse orthographic composition

Now we want to represent each façade as a relief depth field on the orthographic view. The question we addressed is how to compose all data from images domain to this orthographic view. First of all, each input image of the building block is over-segmented into patches using [174]. The patch size is a trade-off between accuracy and robustness. For example, we can choose 700 pixels as the minimum patch size if the input images have a resolution of 640×905 pixels, to favor relatively large patches since the reconstructed 3D points from images are noise.

Algorithm 2 Inverse Orthographic Patching

```
for each image I_k visible to the façade do
    for each super pixel p_i ∈ I_k do
        if normal direction of p_i parallel with z-axis then
            for each pixel (x, y) in the bounding box do
                X ← (x, y, z_i)^T                          ▷ z_i is the depth of p_i
                compute projection (u, v) of X to Camera i
                if super pixel index of (u, v) in I_k = k then
                    accumulate depth z_i, color, segmentation
                end if
            end for
        end if
    end for
end for
```

Patch reconstruction The normal vector and center position of each p_i are estimated from the set of 3D points $\mathcal{P}_i = \{(x_k, y_k, z_k)\}$, which have projections inside p_i. As the local coordinate frame of the block is aligned with the three major orthogonal directions of the building, the computation is straightforward. Let σ_x^i, σ_y^i and σ_z^i be the standard deviations of all 3D points in $\mathcal{P}_i$ in three directions. We first compute the normalized standard deviations $\hat{\sigma}_x^i = \frac{\bar{s}_x}{s_x^i}\sigma_x^i$, $\hat{\sigma}_y^i = \frac{\bar{s}_y}{s_y^i}\sigma_y^i$, where s_x^i and s_y^i are the horizontal and vertical sizes of the bounding box of the patch in the input images, and their median respectively across all patches $\bar{s}_x = \text{median}_i s_x^i$, $\bar{s}_y = \text{median}_i s_y^i$. The normalization avoids bias to a small patch. The patch p_i is regarded as parallel to the façade base plane if σ_z is smaller than σ_x^i and σ_y^i. And, all these parallel patches with small σ_z contribute to the composition of an orthographic view of the façade. The orientation of such a patch p_i is aligned with the z-axis, and its position set at the depth $z_i = \text{median}_{(x_j,y_j,z_j)\in p_i} z_j$. One example is shown in Figure 4.1(a).

Orthographic composition To simplify the representation for irregular shapes of the patches, we deploy a discrete 2D orthographic space on the xy-plane to create an orthographic view O of the façade. The size and position of O on the xy-plane are determined by the bounding box of the 3D points of the block, and the resolution of O is a parameter that is actually set not to exceed 1024×1024. Each patch is mapped from its original image space onto this orthographic space as illustrated in Figure 4.1 from (a) to (b). We use an inverse orthographic mapping algorithm shown in Algorithm 2 to avoid gaps. Theoretically, the warped textures of all patches create a true orthoimage O as each used patch has a known depth and is parallel with the base plane.

For each pixel v_i of the orthoimage O, we accumulate a set of depth values $\{z_j\}$, a set corresponding of color values $\{\mathbf{c}_j\}$ and a set of segmentation labels $\{l_j\}$. The depth of

this pixel is set to the median of $\{z_j\}$ whose index is $\kappa = \arg\text{median}_j z_j$. Since the depth determines the texture color and segmentation label, we take $\mathbf{c}_\kappa$ and l_κ as the estimated color and label for the pixel. In practice, we accept a small set of estimated points around z_κ and take their mean as the color value in the texture composition. As the content of images are highly overlapped, if a pixel is observed only once from one image, it is very likely that it comes from an incorrect reconstruction. It will thus be rejected in the depth fusion process. Moreover, all pixels $\{v_i\}$ with multiple observations $\left\{\{z_j\}_i\right\}$ are sorted in non-decreasing order according to their standard deviation $\varsigma_i = sd\left(\{z_j\}\right)$ of depth sets. After that, we define $\varsigma\left(\eta\right)$ to be the $\eta\left|\{v_i\}\right|$-th element in the sorted $\{\varsigma_i\}$. We declare the pixel v_i to be unreliable if $\varsigma_i > \varsigma\left(\eta\right)$. The value of η comes from the estimated confidence of the depth measurements. We currently scale the value by the ratio of the number of 3D points and the total pixel number of O.

Note that when we reconstruct the patches, we do not use the semantic segmentation results in the input image space for two reasons. The first is that the patches used in reconstruction are much larger in size than those used for semantic segmentation, and this may lead to an inconsistent labeling. Though it is possible to estimate a unique label for a patch, it may downgrade the semantic segmentation accuracy. The second is that the possible errors in the semantic segmentation may over-reject patches, which compromises the quality of the depth estimation. Therefore, we reconstruct the depth first and transfer the segmentation results from the input image space to the orthographic view with pixel-level accuracy, shown in Figure 4.1(f). After that, we remove the non-building pixels in the orthoimage according to the segmentation label. Our composition algorithm for the orthographic depth map is functionally close to the depth map fusion techniques such as [175]. But our technique is robust as we use the architectural prior of orthogonality that preserves structural discontinuity without over-smoothing.

4.1.2 Data structure

The 2.5D relief map representation is a bitmap representation of a façade. In this section, we propose a vector graphics approach, to enable easy editing and rendering. Since we want to represent the façade on a level higher than pixel, we want to decompose it using some larger units. Decomposing a façade is also analyzing the structure in the hope of reconstructing it with a smaller number of elements. In the first approximation, taking all horizontal and vertical lines, and the intersections form a grid that gives an irregular partition of the reference plane into rectangles of various sizes.

This partition captures the global rectilinear structure of the façades and buildings, also keeps all discontinuities of the façade substructures. This usually gives an over-

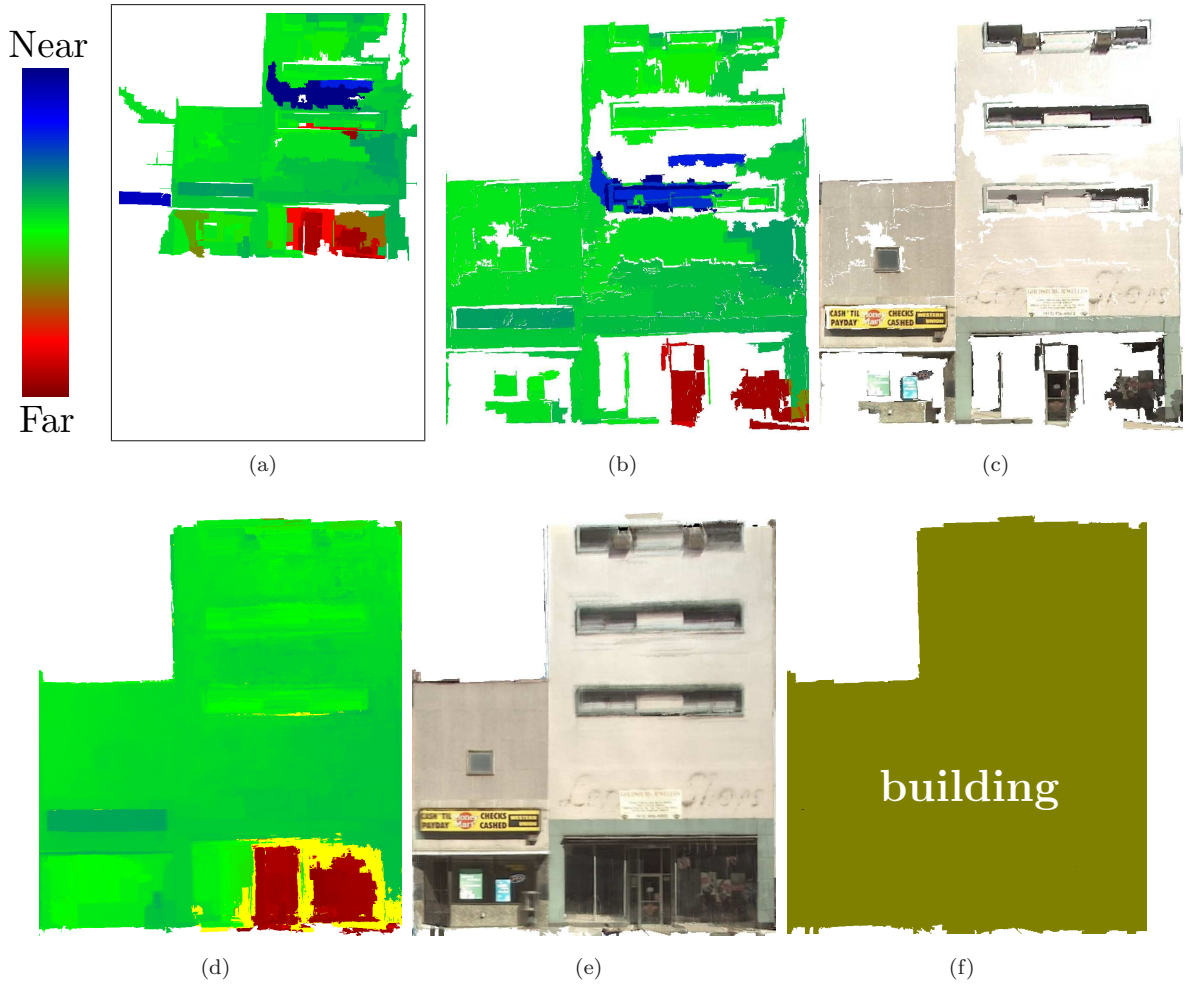


Figure 4.1: **Inverse orthographic composition.** (a) Depth map in input image space. (b) Partial orthographic depth map from one view. (c) Partial orthographic texture from one view. (d) Composed orthographic depth map (unreliably estimated pixels are in yellow). (e) Composed orthographic texture. (f) Composed orthographic building region.

segmentation of the image into patches. But this over-segmentation has several advantages. The over-segmenting lines can also be regarded as auxiliary lines that regulate the compositional units of the façades and buildings. Some "hidden" rectilinear structure of the façade during the construction can also be re-discovered by this over-segmentation process.

Hidden Structure Discovery

To discover the structure inside the façade, the edge of the reference texture image is first detected by [176] because it is very efficient and robust to image noises empirically. With such edge maps, hough transformation [177] is used to recover the line structure. To improve robustness, the direction of hough transformation is constrained to only horizontal and vertical, which happens in most situations of architecture façade. The detected lines

are now formed a grid to partition the whole reference image, and this grid contains many non-overlapped small line segments by taking intersection of hough lines as endpoints as in Figure 4.3(b).

These small line segments are now the hypothesis to partition the façade. The hough transformation is good for structure discovery since it can extract the hidden global information from the façade and align small line segments to this hidden structure. However, some small line segments in the formed grid may not really be a partition boundary between different regions. Hence, a weight w_e is defined on each small line segment e to indicate the likelihood that this small line segment is a boundary of two different region as shown in Figure 4.3(c). This weight is computed as the number of edge point from Canny edge map that the line segment covered.

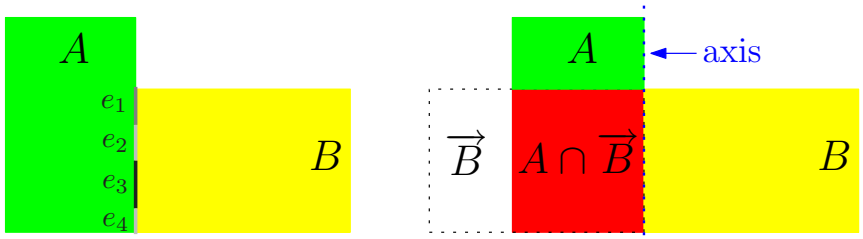
Remark on over-segmented partition It is true that the current partition schema is subject to segmentation parameters. But it is important to note that usually a slightly over-segmented partition is not harmful for the purpose of modeling. A perfect partition certainly eases the regularization of the façade augmentation by the depths presented in the next section, nevertheless, an imperfect, particularly a slight over-segmented partition, does not affect the modeling results particularly even the 3D points are dense and the stereo works well.

Recursive Subdivision

Given a region D in the reference texture image, it is divided into two sub rectangle regions D_1 and D_2, such that $D = D_1 \cup D_2$, by a line segment L with strongest support from the edge points. After D is subdivided into two separate regions, the subdivision procedures continue on the two regions D_1 and D_2 recursively. The recursive subdivision procedure is stopped if either the target region D is too small to be subdivided, or their is no strong enough hypothesis, i.e. the region D is very smooth.

For street-view, the bilateral symmetry about a vertical axis may not exist for the whole façade, but it exists locally and can be used for more robust subdivision. First, for each region D, the NCC score s_D of the two halves D_1 and D_2 vertically divided at the center of D is computed. If $s_D > \eta$, region D is considered to have the bilateral symmetry. Then, the edge map of D_1 and D_2 are averaged, and subdivision is recursively done on D_1 only. Finally, the subdivision in D_1 is reflected across the axis to become the subdivision of D_2, and merged the two subdivisions into the subdivision of D.

Recursive subdivision is good to preserve boundaries for man-made structure styles. However, it may produce some unnecessary fragments for depth computation and render-



(a) Edge Weight Support (b) Region Statistic Support

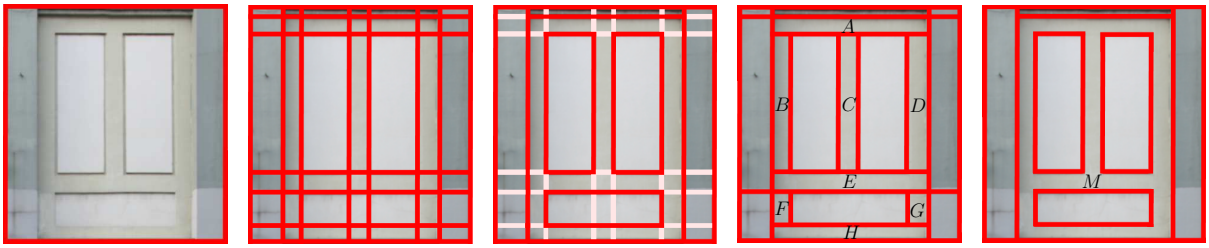
$$s_{AB} = \max\{w_{e_1}, w_{e_2}, w_{e_3}, w_{e_4}\} + \kappa \mathrm{NCC}(A \cap \overrightarrow{B}, \overleftarrow{\overleftarrow{A} \cap B})$$

Figure 4.2: **Merging support evaluation.**

ing as in Figure 4.3(d). Hence, as a post-processing, if two leaf subdivision regions are neighbors, and there is enough support to separate them, they are merged into one region. The support s_{AB} to separate two neighbor regions A and B is defined to be the strongest weights of all the small line segments on the border between A and B: $s_{AB} = \max_e \{w_e\}$. However, the weights of small line segments can only offer a local image statistic on the border. To improve the robustness, as a dual information region statistic between A and B can be used more globally. As in Figure 4.2, Since regions A and B may not have the same size, this region statistic similarity is defined as the following: First an axis is defined on the border between A and B and region B is reflected on this axis to have a region $\overrightarrow{B}$. The overlapped region $A \cap \overrightarrow{B}$ between A and $\overrightarrow{B}$ is defined to be the pixels from A with location inside $\overrightarrow{B}$. In a similar way, $\overleftarrow{A} \cap B$ contains the pixels from B with location inside $\overleftarrow{A}$, and then it is reflected to become $\overleftarrow{\overleftarrow{A} \cap B}$ according the same axis. The normalized cross correlation (NCC) between $A \cap \overrightarrow{B}$ and $\overleftarrow{\overleftarrow{A} \cap B}$ is used to defined the region similarity of A and B. In this way, only the symmetric part of A and B is used for region comparison. Therefore, the affect of other far away part of the region is avoided, which will happen if the size of A and B is dramatically different and global statistic such as color histogram is used. Weighted by a parameter κ, the support s_{AB} to separate two neighbor regions A and B is now defined as

$$s_{AB} = \max_e \{w_e\} + \kappa \mathrm{NCC}(A \cap \overrightarrow{B}, \overleftarrow{\overleftarrow{A} \cap B}).$$

Notice that the representation of the façade is a binary recursive tree before merging. And now after region merging, it is a Directed Acyclic Graph (DAG). The DAG representation can innately support Level of Detail displaying technique. When great details are demanded, the rendering engine can just go down the rendering graph to expand all detail leaves and render them correspondingly. Vice verse, the intermediate node is rendered and all descendent are pruned at rendering time.



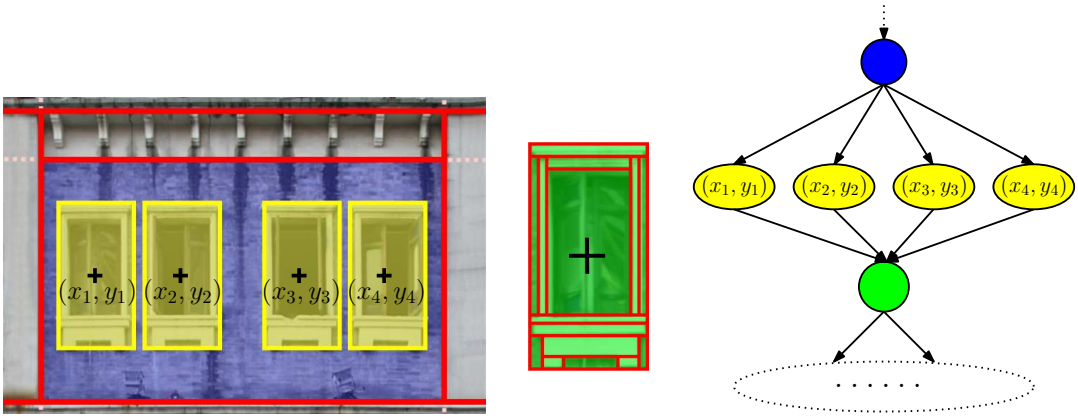
(a) Input (b) Structure (c) Weight (d) Subdivide (e) Merge

Figure 4.3: **Structure preserving subdivision.** The hidden structure of the façade is extracted out to form a grid in (b). Such hypothesis is evaluated according to the edge support in (c), and the façade is recursive subdivided into several regions in (d). Since there are no enough supports between Region A, B, C, D, E, F, G, H, they are all merged into one single region M in (e).

Repetitive Pattern Representation

Repeat pattern of façade exists globally for higher floors of building, but not in the ground floors that we are mainly working on. In [178], the essential step, to detect the global repetitive pattern and to use them for analysis of the façade structure, is not applicable in our street-view modeling. In our workflow, this step is replaced by the recursive subdivision described in the previous subsection. However, the repetitive pattern of the façade does locally exist in many façades and most of them are windows. [178] uses a quite complicated technique for synchronization of subdivision between different windows.

To save storage space and easy the synchronization task, in our method, only one subdivision representation for the same types of windows are maintained. Precisely, window template is first detected by trained model [179] or manually indication on the reference texture images. The templates are matched across the reference texture image using NCC as measurement. If good enough matchings exist, they are firstly aligned according to the horizontal or vertical direction by a hierarchical clustering, and the canny edge maps on these regions are averaged. During the subdivision, each matched region is isolated by shrinking a bounding rectangle on the average edge maps until snapping to strong edge, and is regarded as a whole leaf region. The edges inside these isolated regions should not affect the global structure, and hence these edge points are not used during the global subdivision procedure. Then, as in Figure 4.4, all the matched leaf regions are linked to the root of a common subdivision DAG for that type of window, by introducing a 2D translation node for the pivot position. Recursive subdivision is again run on the average edge maps of all matched regions. To preserve photo realism, the textures in these regions are not shared and only the subdivision DAG and their respective depths are shared. Fur-



(a) Façade (b) DAG

Figure 4.4: **A DAG for repetitive pattern representation.**

thermore, to improve the robustness of subdivision, vertical bilateral symmetric is taken as hard constraint for windows.

4.2 Structure analysis

From the composed orthographic depth map and texture image for each façade, we want to identify the structural elements at different depths of the façade to enrich the façade geometry. To cope with the irregular, noisy and missing depth estimations on the façade, a strong regularization from the architecture priors is therefore required. Most of buildings are governed by vertical and horizontal lines and form naturally rectangular shapes. We restrict the prior shape of each distinct structure element to be a rectangle, such as the typical extruding signboard in Figure 4.1.

4.2.1 Joint analysis

We use a bottom-up, graph-based segmentation framework [174] to jointly segment the orthographic texture and depth maps into regions, where each region is considered as a distinct element within the façade. The proposed shape-based segmentation method jointly utilizes texture and depth information, and enables the fully automatic façade structure analysis. Xiao *et al.* [64] also proposed a functional equivalent top-down recursive subdivision method. However, it has been shown in [64] to be inefficient to produce satisfiable result without any user interaction.

A graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ is defined on the orthoimage image O with all pixels as vertices $\mathcal{V}$ and edges $\mathcal{E}$ connecting neighboring pixels. To encourage horizontal and vertical cut, we use 4-neighborhood system to construct $\mathcal{E}$. The weight function for an edge connecting two

pixels with reliable depth estimations is based both on the color distance and normalized depth difference

$$w\left(\left(v_i, v_j\right)\right) = \left\|\mathbf{c}_i - \mathbf{c}_j\right\|^2 \cdot \left(\frac{z_i - z_j}{\varsigma\left(\eta\right)}\right)^2,$$

where $\left\|\mathbf{c}_i - \mathbf{c}_j\right\|^2$ is the L_2-Norm of the RGB color difference of two pixels v_i and v_j. We slightly pre-filter the texture image using a Gaussian of small variance before computing the edge weights. The weight for an edge connecting two pixels without reliable depth estimations is set to 0 to force them to have the same label. We do not construct an edge between a pixel with a reliable depth and a pixel without a reliable depth, as the weight cannot be defined.

We first sort $\mathcal{E}$ by non-decreasing edge weight w. Starting with an initial segmentation in which each vertex v_i is in its own component, the algorithm repeats for each edge $e_q = (v_i, v_j)$ in order for the following process: If v_i and v_j are in disjoint components $C_i \neq C_j$, and $w\left(e_q\right)$ is small compared with the internal difference of both those components, $w\left(e_q\right) \leq MInt\left(C_i, C_j\right)$, then the two components are merged. The minimum internal difference is defined as

$$MInt\left(C_1, C_2\right) = \min\left(Int\left(C_1\right) + \tau\left(C_1\right), Int\left(C_2\right) + \tau\left(C_2\right)\right),$$

where the internal difference of a component C is the largest weight in the minimum spanning tree of the component

$$Int\left(C\right) = \max_{e \in MST(C,E)} w\left(e\right).$$

The non-negative threshold function $\tau\left(C\right)$ is defined on each component C. The difference in this threshold function between two components must be greater than their internal difference for an evidence of a boundary between them. Since we favor a rectangular shape for each region, the threshold function $\tau\left(C\right)$ is defined by the divergence $\vartheta\left(C\right)$ between the component C and a rectangle, which is the portion of the bounding box B_C with respect to the component C, $\vartheta\left(C\right) = \left|B_C\right| / \left|C\right|$. For small components, $Int\left(C\right)$ is not a good estimate of the local characteristics of the data. Therefore, we let the threshold function be adaptive based on the component size,

$$\tau\left(C\right) = \left(\frac{\varrho}{\left|C\right|}\right)^{\vartheta(C)},$$

where ϱ is a constant and is set to 3.2 in our prototype. τ is large for components that do not fit a rectangle, and two components with large τ are more likely to be merged. A larger ϱ favors larger components, as we require stronger evidence of a boundary for smaller components.

Once the segmentation is accomplished, the depth values for all pixels in C_i of each reliable component C_i are set to the median. The depth of the largest region is regarded as the depth of the base plane for the façade. Moreover, an unreliable component C_i smaller than a particular size, set to 4% of the current façade area, is merged to its only reliable neighboring component if such a neighboring component exists.

4.2.2 Shape regularization

Except for the base plane of the façade, we fit a rectangle to each element on the façade. For an element $C = \{v_i = (x_i, y_i)\}$, we first obtain the median position $(x_{\text{med}}, y_{\text{med}})$ by $x_{\text{med}} = \text{median}_i x_i$ and $y_{\text{med}} = \text{median}_i y_i$. We then remove outlier points that are $|x_i - x_{\text{med}}| > 2.8\sigma_x$ or $|y_i - y_{\text{med}}| > 2.8\sigma_y$, where $\sigma_x = \sum_i |x_i - x_{\text{med}}| / |C|$ and $\sigma_y = \sum_i |y_i - y_{\text{med}}| / |C|$. Furthermore, we reject the points that are in the 1% region of the left, right, top and bottom according to their ranking of x and y coordinates in the remaining point set. In this way, we obtain a reliable subset C_{sub} of C. We define the bounding box $B_{C_{\text{sub}}}$ of C_{sub} as the fitting rectangle of C. The fitting confidence is then defined as

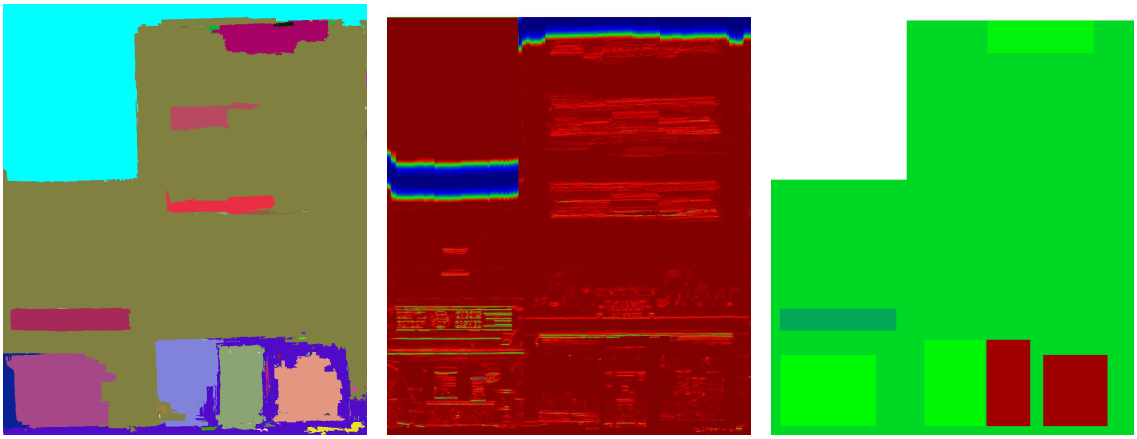
$$f_C = \frac{B_{C_{\text{sub}}} \cap C}{B_{C_{\text{sub}}} \cup C}.$$

In the end, we only retain the rectangles as distinct façade elements if the confidence $f_C > 0.72$ and the rectangle size is not too small.

The rectangular elements are automatically snapped into the nearest vertical and horizontal mode positions of the accumulated Sobel responses on the composed texture image, if their distances are less than 2% of the width and height of the current façade. The detected rectangles can be nested within each other. When producing the final 3D model, we first pop up the larger element from the base plane and then the smaller element within the larger element. If two rectangles overlap but do not contain each other, we first pop up the one that is closest to the base plane.

4.2.3 Repetitive pattern rediscovery

Structure elements are automatically reconstructed in the previous section. However, when the depth composition quality is not good enough due to poor image matching, reflective materials or low image quality, only a few of them could be successfully recovered. For repetitive elements of the façade, we can now systematically launch a re-discovery process using the discovered elements as templates in the orthographic texture image domain. The idea of taking advantage of repetitive nature of the elements has been explored in [178, 64].



(a) (b) (c)

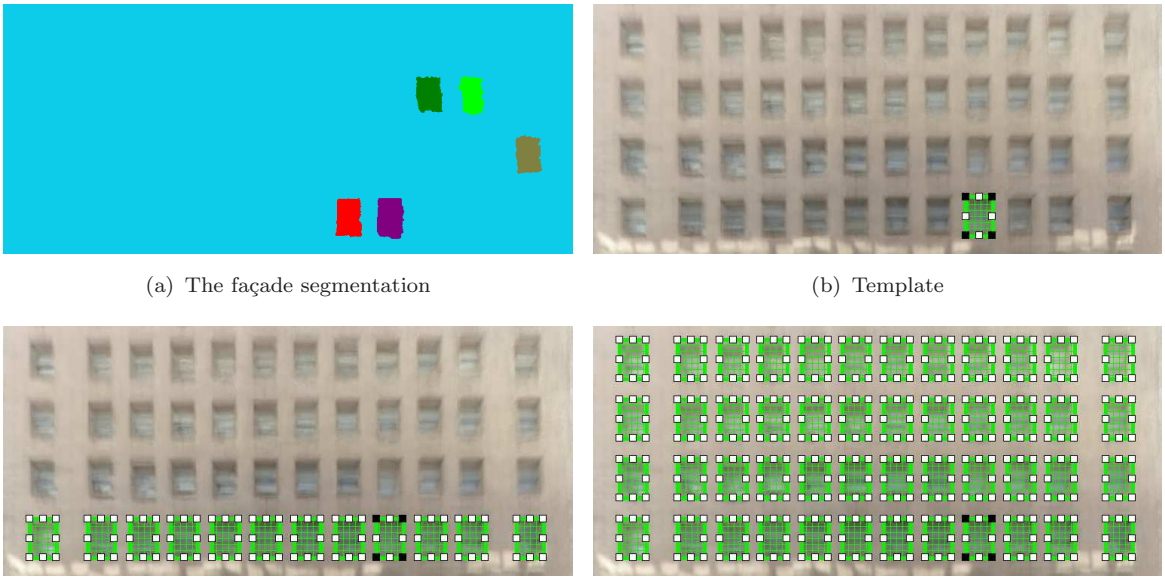
Figure 4.5: **Structure analysis and regularization for modeling.** (a) The façade segmentation. (b) The data cost of boundary regularization. The cost is color-coded from high at red to low at blue via green as the middle. (c) The regularized depth map.

We use the Sum of Squared Differences (SSD) on RGB channels for template matching. Unlike [64] operating in 2D search space, we use a two-step method to search twice in 1D, shown in Figure 4.6. We first search in horizontal direction for a template B_i and obtain a set of matches $\mathbf{B}_i$ by extracting the local minima under a threshold. Then, we use both B_i and $\mathbf{B}_i$ together as the template to search for the local minima along the vertical direction. This leads to more efficient and robust matching, and automatic alignment of the elements. A re-discovered element by template matching inherits the depth of the template.

When there are more than one structure element discovered previously by joint segmentation representing the same kind of structure elements, we also need to cluster the re-discovered elements using a bottom-up hierarchical merging mechanism. Two templates B_i and B_j obtained by joint segmentation with sets of matching candidates $\mathcal{M}_i$ and $\mathcal{M}_j$ are merged into the same class, if one template is sufficiently similar to any element of the candidates of the other template. Here, the similarity between two elements is defined as the ratio of the intersection area by the union area of the two elements. The merging process consists of averaging element sizes between $\mathcal{M}_i \cup \{B_i\}$ and $\mathcal{M}_j \cup \{B_j\}$, as well as computing the average positions for overlapped elements in $\mathcal{M}_i \cup \{B_i\}$ and $\mathcal{M}_j \cup \{B_j\}$.

4.3 Boundary regularization

The boundaries of the façade of a block are further regularized to favor sharp change and penalize serration. We use the same method as for shape regularization of structure



(a) The façade segmentation (b) Template

(c) Horizontal matching (d) Matching results

Figure 4.6: **Repetitive pattern rediscovery.**

elements to compute the bounding box $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$ of the façade. Finally, we further optimize the upper boundary of the façade, as we cannot guarantee that a building block is indeed a single building with the same height during block partition.

Illustrated in Figure 4.7, we lay out a 1D Markov random field on the horizontal direction of the orthoimage. Each $x_i \in [x_{\min}, x_{\max}]$ defines a vertex, and an edge is added for two neighboring vertices. The label l_i of x_i corresponds to the position of the boundary, and $l_i \in [y_{\min}, y_{\max}]$ for all x_i. Therefore, one label configuration of the MRF corresponds to one façade boundary. Now, we utilize all texture, depth and segmentation information to define the cost.

The data cost is defined according to the horizontal Sobel responses

$$\phi_i(l_j) = 1 - \frac{\text{HorizontalSobel}(i,j)}{2\max_{xy}\text{HorizontalSobel}(x,y)}.$$

Furthermore, if l_j is close to the top boundary r_i of reliable depth map, $|l_j - r_i| < \beta$, where β is empirically set to $0.05(y_{\max} - y_{\min} + 1)$, we update the cost by multiplying it with $(|l_j - r_i| + \epsilon)/(\beta + \epsilon)$. Similarly, if l_j is close to the top boundary s_i of segmentation $|l_j - s_i| < \beta$, we update the cost by multiplying it with $(|l_j - s_i| + \epsilon)/(\beta + \epsilon)$. For the façades whose boundaries are not in the viewing field of any input image, we snap the façade boundary to the top boundary of the bounding box, and empirically update $\phi_i(y_{\min})$ by multiplying it with 0.8. Figure 4.5(b) shows one example of defined data cost.

The height of the façade upper boundary usually changes in the regions with strong vertical edge responses. We thus accumulate vertical Sobel responses at each x_i into

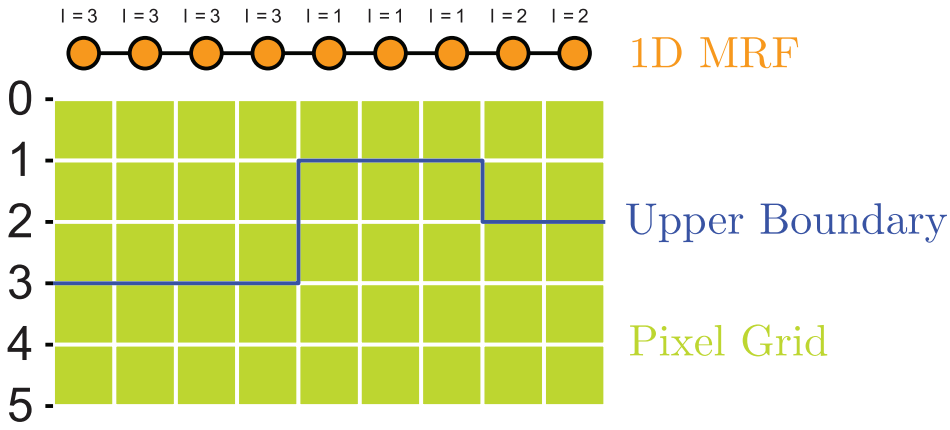


Figure 4.7: **An example of MRF to optimize façade upper boundary.**

$V_i = \sum_{y \in [y_{\min}, y_{\max}]} \text{VerSobel}(i, y)$, and define the smoothness term to be

$$\phi_{i,i+1}(l_i, l_{i+1}) = \mu |l_i - l_{i+1}| \left(1 - \frac{V_i + V_{i+1}}{2 \max_j V_j}\right),$$

where μ is a controllable parameter.

The boundary is optimized by minimizing a Gibbs energy [158]

$$E(L) = \sum_{x_i \in [x_{\min}, x_{\max}]} \phi_i(l_i) + \sum_{x_i \in [x_{\min}, x_{\max}-1]} \phi_{i,i+1}(l_i, l_{i+1}),$$

where ϕ_i is the data cost and $\phi_{i,i+1}$ is the smoothing cost. The exact inference can be obtained with a global optimum by methods such as belief propagation [180].

4.4 Texture mapping

4.4.1 Model production

Each façade is the front side of the building block. We can extend a façade in the z-direction into a box with a constant depth (the default constant is set to 18 meters in the current implementation) to represent the geometry of the building block, as illustrated in Figure 4.4.1(c).

All the blocks of a sequence are then assembled into the street side model. The texture mapping is done by visibility checking using z-buffer ordering. The side face of each block can be automatically textured as illustrated in Figure 4.4.1 if it is not blocked by the neighboring buildings.

(a) (b) (c)

Figure 4.8: **Model production and texture mapping.** (a) The texture-mapped façade. (b) The texture-mapped block. (c) The block geometry.

4.4.2 Occlusion removal

The geometry of the façade is initialized as a flat rectangle. Usually a façade is too big to be entirely observable in one input image. We first compose a texture image for the entire rectangle of the façade from the visible images of the façade. This process is different from image mosaic, as the images have parallax that is helpful for removing the undesired occluding objects, such as pedestrians, cars, trees, telegraph poles and rubbish bins, in front of the target façade. Since we have a known position initial plane for the target façade, occluders can be removed by photo consistency checking to obtain a better texture image than pure mosaic.

As many multi-view stereo methods [3], the photo consistency is defined as followed. Consider a 3D point $\mathbf{X} = (x, y, z, 1)'$ with color $\mathbf{c}$. If it has a projection $\mathbf{x}_i = (u_i, v_i, 1)' = \mathbf{P}_i\mathbf{X}$ in the i-th camera $\mathbf{P}_i$, under Lambertian surface assumption, the projection $\mathbf{x}_i$ should also have the same color $\mathbf{c}$. However, if the point is occluded by some other objects in this camera, the color of the projection is usually not the same as $\mathbf{c}$. Note that $\mathbf{c}$ is unknown and what we want here. Assuming the point $\mathbf{X}$ is visible from multiple cameras $\mathcal{I} = \{\mathbf{P}_i\}$ and occluded by some objects in the other cameras $\mathcal{I}' = \{\mathbf{P}_j\}$, then the color $\mathbf{c}_i$ of the projections in $\mathcal{I}$ should be the same as $\mathbf{c}$, while it may be different from the color $\mathbf{c}_j$ of projections in $\mathcal{I}'$. Now, given the a set of projection color $\{\mathbf{c}_k\}$, the task is to identify a set $\mathcal{O}$ of the occluded cameras. In most of the situation, we can assume that the point $\mathbf{X}$ is visible from most of the cameras. Under this assumption, we have $\widehat{\mathbf{c}} \approx \text{median}_k \{\mathbf{c}_k\}$. Given the estimated color of the 3D point $\widehat{\mathbf{c}}$, it is now very easy to identify the occluded set $\mathcal{O}$ according to their distances with $\widehat{\mathbf{c}}$. To improve the robustness, instead of a single color,

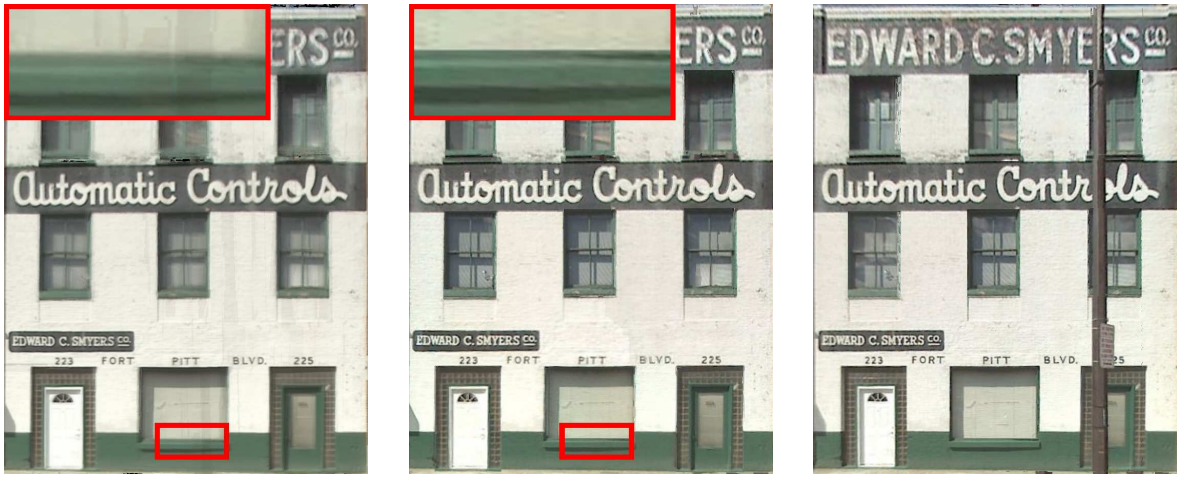
Algorithm 3 Photo Consistency Check For Occlusion Detection

Require: A set of N image patches $\mathcal{P} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_N\}$ corresponding to the projections of the 3D point $\mathbf{X}$.
Require: $\eta \in [0, 1]$ to indicate when two patches are similar.

for all $\mathbf{p}_i \in \mathcal{P}$ **do**
  $s_i \leftarrow 0$ ▷ Accumulated similarity for $\mathbf{p}_i$
  for all $\mathbf{p}_j \in \mathcal{P}$ **do**
    $s_{ij} \leftarrow \mathrm{NCC}(\mathbf{p}_i, \mathbf{p}_j)$
    if $s_{ij} > \eta$ **then** $s_i \leftarrow s_i + s_{ij}$
    end if
  end for
end for
$\widehat{n} \leftarrow \arg\max_i s_i$ ▷ $\widehat{n}$ is the patch with best support
$\mathcal{V} \leftarrow \emptyset$ ▷ $\mathcal{V}$ is the index set with visible projection
$\mathcal{O} \leftarrow \emptyset$ ▷ $\mathcal{V}$ is the index set with occluded projection
for all $\mathbf{p}_i \in \mathcal{P}$ **do**
  if $s_{i\widehat{n}} > \eta$ **then** $\mathcal{V} \leftarrow \mathcal{V} \cup \{i\}$
  else $\mathcal{O} \leftarrow \mathcal{O} \cup \{i\}$
  end if
end for
return $\mathcal{V}$ and $\mathcal{O}$

the image patches centered at the projections are used, and patch similarity, normalized cross correlation (NCC), is used as a metric. The detail is presented in Alg. 3. In this way, with the assumption that the façade is almost planar, each pixel of the reference texture corresponds to a point lies on the flat façade. Hence, for each pixel, we can identify whether it is occluded in a particular camera.

Now, for a given planar façade in space, all visible images are sorted according to the fronto-parallelism of the images with respect to the given façade. An image is to be more fronto-parallel if the projected surface of the façade in the image is larger. The reference image is first warped from the most fronto-parallel image, then from the lesser ones according to the visibility of the point. In each step, because the occluding region by other objects is not pasted on the reference image, some region of the reference texture image still leaves empty. In a later step, if that empty region is not occluded and visible from the new camera, the region is filled. In this way, a multi-view inpainting is done to fill the occluded region from each single camera. At the end of the process, if some regions are still empty, a normal image inpainting technique are used to fill it either automatically by [181] or interactively with guide by the users in Section 4.5.1. Since we have adjusted the cameras according to the image correspondence, this simple mosaic without explicit blending can already produce very visual pleasing results.



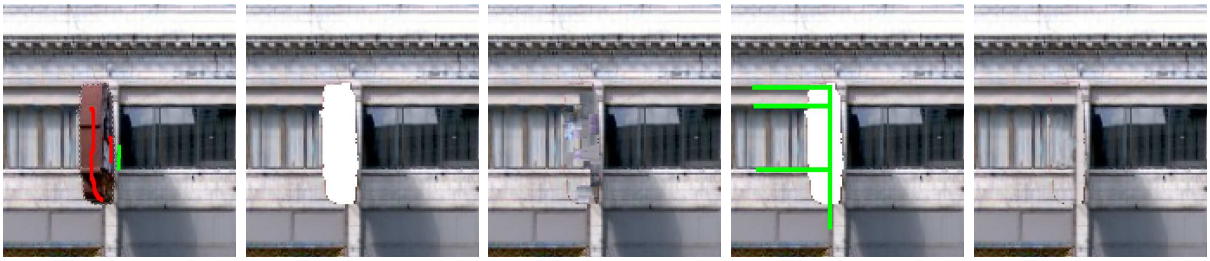
(a) (b) (c)

Figure 4.9: **Texture optimization.** (a) The original orthographic texture image. (b) The optimized texture image. (c) A direct texture composition. The optimized texture image in (b) is more clear than the original orthographic texture image in (a), and has no texture from occluding objects, such as the one contained in (c).

4.4.3 Texture optimization

The orthographic texture for each front façade by Algorithm 1 is a true orthographic texture map. But as a texture image, it suffers from the artifacts of color discontinuities, blur and gaps, as each pixel has been independently computed as the color of the median depth from all visible views. However, it does provide very robust and reliable information for the true texture, and contain almost no outlier from occluding objects. Therefore, we re-compute an optimized texture image for each front façade, regarding the original orthographic texture image as a good reference.

Suppose that each façade has N visible views. Each visible view is used to compute a partial texture image for all visible points of the façade. Then we obtained N partial texture images for the façade. Next, we define a difference measurement as the squared sum of differences between each pixel of the partial texture images and the original orthographic texture image at the same coordinate. This is the data term for a Markov Random Field on the orthographic texture image grid. The smoothing term is defined to be the reciprocal of the color difference between each neighboring pair of pixels on the original orthographic texture image. The desired orthographic texture image is computed using GraphCut alpha-expansion [159]. If seam artifacts are serious, poisson blending [182] can be used as post-process. Figure 4.9 shows the comparative results of this process. Figure 4.9(c) also shows a direct texture warping from most fronto-parallel image as in [64], which fails to remove the occluding objects, i.e. the telegraph pole in this case.



(a) Indicate (b) Remove (c) Inpaint (d) Guide (e) Result

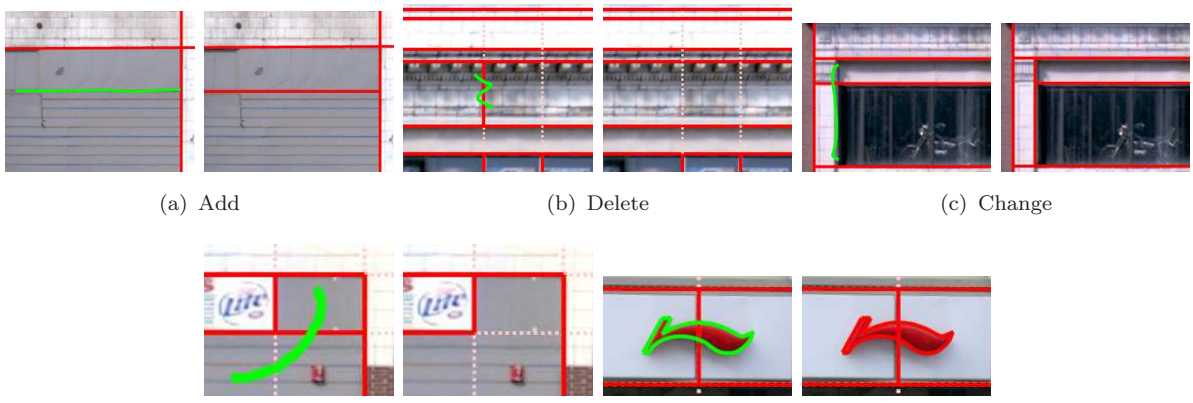
Figure 4.10: **Interactive texture refinement.** (a) The user draws strokes on the advertisement billboard to indicate removal. (b) The billboard is removed. (c) The region is automatically inpainted. (d) If not satisfying, some lines (in green) can be drawn to guide the structure. (e) Now better result is achieved with the guide lines.

4.5 Editing

So far, the modeling is computed fully-automatically. If the user wants to make some manual efforts for improvement, we also provide several convenient mechanisms for model editing.

4.5.1 Interactive texture inpainting

As shown in Figure 4.10, if the automatic texture composition result is not satisfactory, a two-step interactive user interface is provided for refinement. In the first step, the user can draw strokes to indicate which object or part of the texture is undesirable as in Figure 4.10(a). The corresponding region is automatically extracted based on the input strokes as in Figure 4.10(b) using the method in [183]. The removal operation can be interpreted as that the most fronto-parallel and photo-consisted texture selection, from the result of Algorithm 3, is not what the user wants. For each pixel, $\widehat{n}$ from Line 9 of Alg. 3 and $\mathcal{V}$ should be wrong. Hence, $\mathcal{P}$ is updated to excluded $\mathcal{V}$: $\mathcal{P} \leftarrow \mathcal{O}$. Then, if $\mathcal{P} \neq \emptyset$, Alg. 3 is run again. Otherwise, image inpainting [181] is used for automatically inpainting as in Figure 4.10(c). At the second step, if the automatic texture filling is poor, the user can manually specifies important missing structure information by extending a few curves or line segments from the known to the unknown regions as in Figure 4.10(d). Then as in [184], image patches are synthesized along these user-specified curves in the unknown region using patches selected around the curves in the known region by Loopy Belief Propagation to find the optimal patches. After completing structure propagation, the remaining unknown regions are filled using patch-based texture synthesis as in Figure 4.10(e).



(a) Add (b) Delete (c) Change

(d) Group (e) Carve

Figure 4.11: **Five operations for subdivision refinement.** In each subfigure, the left figure is the original subdivision layout shown in red and the user sketched stroke shown in green, while the right figure is the result subdivision layout.

4.5.2 Interactive subdivision refinement

In most situations, the automatic subdivision works satisfactorily. If the user wants to further refine the subdivision layout, three line operations and two region operations are provided as hard constraints as in Figure 4.11. On the other hand, the current automatic subdivision operates on the horizontal and vertical directions for robustness and simplicity. The fifth 'carve' operator allows the user to manually sketch arbitrarily shaped objects, appeared less frequently, to be included in the façade representation.

Add To partition an existing subdivision region, the user can sketch a stroke to indicate the partition as in Figure 4.11(a). The edge points near the stroke are forced to become salient, and hence the subdivision engine can figure the line segment out and partition the region.

Delete To delete a line segment between two regions, the user can sketch a "Z" shape stroke to cross out the line segments as in Figure 4.11(b).

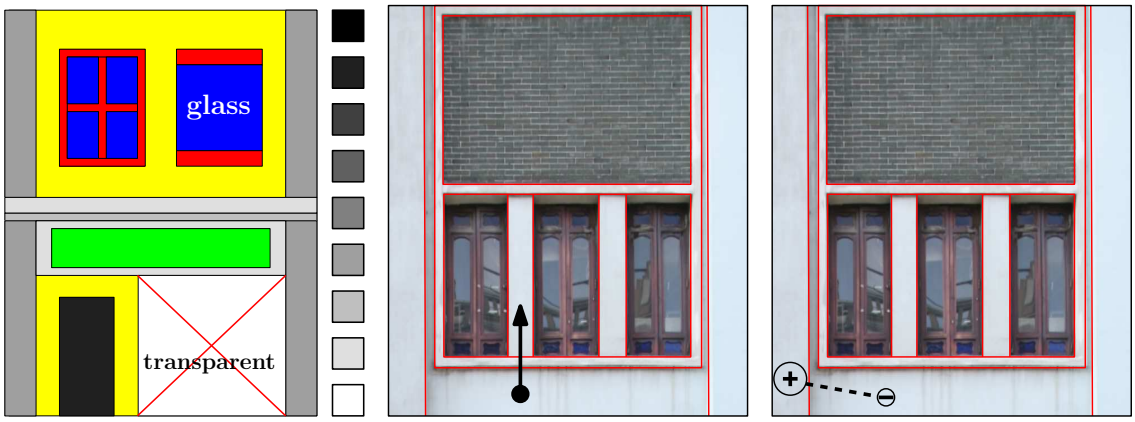
Change To differently partition a region, the user can first delete the partition line segments and then add a new line segment. Alternatively, the user can directly sketch a stroke. Then, the line segment across by the stroke will be deleted and a new line segment will be constructed accordingly as in Figure 4.11(c). After the corresponding operation executed, all descendants with the target region as root node in the DAG representation will be triggered to be re-computed.

Group The user can draw a stroke to cover several regions, in order to merge them into a single group as in Figure 4.11(d).

Carve The user can draw line segments or Non-Uniform Rational B-Spline (NURBS) curve to carve and split the existing subdivision layout as in Figure 4.11(e). In this way, any shape can be extruded and hence be supported.

4.5.3 Interactive depth assignment

In most situations, the depth automatically reconstructed is already good enough for visual inspection. For building that needs more details and better depth displacement such as landmarks, our workflow also provides friendly user interface to facilitate the interactive depth assignment tasks, where rather than 3D interaction, 2D operations are emphasized. All our tools provide interactive feedback and the user can directly assert the result in 3D.



(a) Depth Palettes (b) Transfer (c) Relative Constraint

Figure 4.12: **Three interactive depth assignment operations.** (a) getting depth directly from depth palettes, (a) transferring depth from other region, (b) adding relative constraint.

Transfer from Other Region If it is not easy to directly paint the corresponding depth, the depth can be transferred from other region by dragging an arrow line to indicate the source and target regions.

Relative Depth Constraint The relative depth between two regions can also be constrained by drag a two-circle ended line. The sign symbols in the circles indicate the order, and the radius of the circles, controlling by the + and - key in the keyboard, represent the depth difference. The difference is taken as hard constraint in the MRF optimization by merging the two nodes in the layout into one and updating the data and smooth costs accordingly.

4.5.4 Re-modeling

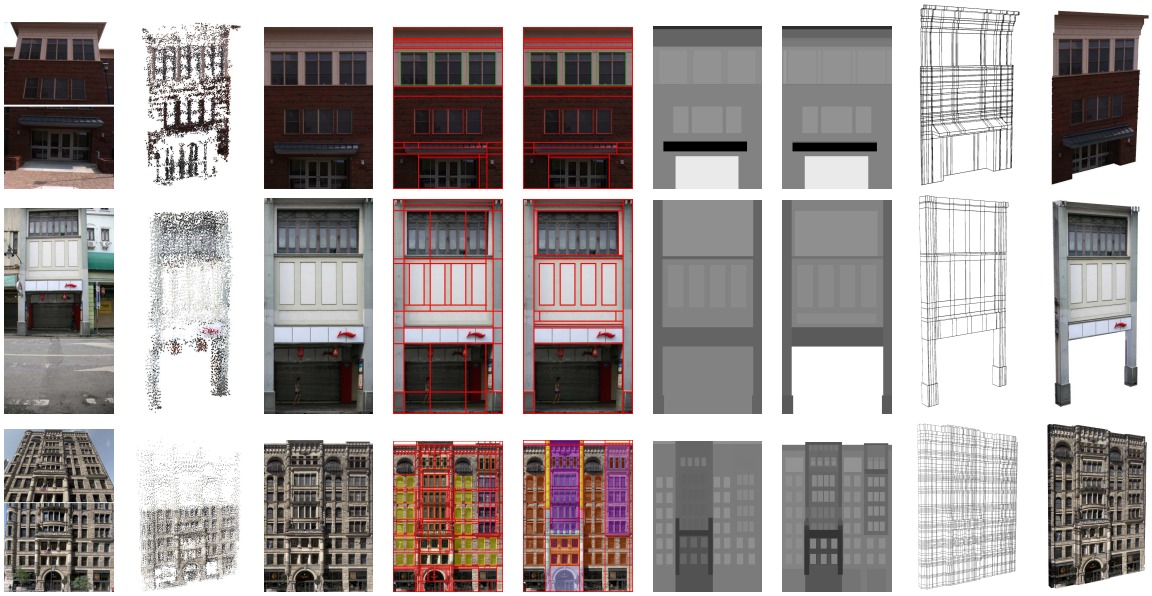
So far, we approximated each elementary unit of the façade as a cubical box that is sufficient for majority of the architectural objects at the scale in which we are interested. Obviously, some façades may have elements of different geometries. Each element can be manually re-modeled by using a pre-defined generic model of type cylinder, sphere, and polygonal solid to replace the given object. The texture is then re-computed automatically from the original images. The columns, the arches, and pediments can be modeled this way. Our decomposition approach makes this replacement convenient particularly for a group of elements with automatic texture re-mapping.

CHAPTER 5

EVALUATION

5.1 Interactive modeling

Three representative large-scale data sets captured under different conditions were chosen to show the flexibility of our approach. Video cameras on a vehicle are used for the first data set, a digital camera on a vehicle for the second, and a handheld camera for the third.



(a) (b) (c) (d) (e) (f) (g) (h) (i)

Figure 5.1: **Two typical façade examples in the first two rows from different data sets and the most difficult example in the third row.** (a) One input view. (b) The 3D points from SFM. (c) The initial textured flat façade. (d) The automatic façade partition (the group of repetitive patterns is color-coded). (e) The user-refined final partition. (f) The re-estimated smoothed façade depth. (g) The user-refined final depth map. (h) The façade geometry. (i) The textured façade model.

For computation of the structure from motion, long sequences were broken down into sub-sequences of about 50 images that are downsampled to the resolution of below 1000×1000. Semi-dense SFM is automatically computed for each subsequence with auto-calibration in about 10 minutes with a PC (CPU Intel Core 2 6400 at 2.13GHz and 3GB

RAM). The subsequences are merged into a global sequence using one fifth of the reconstructed points from the sub-sequences and the GPS/INS (Inertial Navigation System) data if it is available. To capture tall buildings in full, an additional camera captures views looking upwards in 45 degrees, with little or no overlapping between the viewing fields of the cameras. The cameras are mounted on a rigid rig that can be pre-calibrated, so that viewing positions could be transferable between the cameras if the computation for one camera is difficult.

Baity Hill Drive, Chapel Hill. Shown in Figure 5.2, these images were captured by two video cameras [10] of resolution 1024×768 mounted on a vehicle with a GPS/INS. We sampled a sequence of 308 images from each camera. The resulting clouds of 3D points were georegistered with the GPS/INS data. The video image quality is mediocre, although the richness of the building texture is excellent for SFM. It took about 20 minutes for the segmentation on the ground. The geometry of the building blocks is rather simple, and it was reconstructed in about one hour.

Figure 5.2: **The modeling of a Chapel Hill street from 616 images**. Two input images are on the top left; the recovered model rendered is in the bottom row; and two zoomed sections of the recovered model rendered are in the middle and on the right of the top row. The data set is provided by University of North Carolina at Chapel Hill and University of Kentucky [10].

Figure 5.3: **A few façade modeling examples from the two sides of a street with 614 captured images.** Some input images are in the bottom row; the recovered model rendered is in the middle row; and three zoomed sections of the recovered model rendered are in the top row.

Dishifu Road, Canton. Shown in Figure 5.3, these images were captured by a hand-held Canon 5D camera using a 24mm wide lens with 2912×4368 image resolution. A total of 343 views were captured for one side of the street and 271 views for the opposite side.The arcade is modeled using two façades: a front façade for the top part of the arcade and the columns in the front, and a back façade for its lower part. These two façades are merged together and the invisible ceiling connecting the two parts is manually added. The interactive segmentation on the ground took about one hour and the reconstruction took about two hours for both sides of the street.

Hennepin Avenue, Minneapolis. Shown in Figure 5.4, these images were captured by a set of cameras mounted on a vehicle equipped with a GPS/INS system. Each camera has a resolution of 1024×360. The main portion of the Hennepin avenue was covered by a sequence of 130 views using only one of the side-looking cameras. An additional sequence of seven viewing positions taken by an additional side camera pointed 45 degrees up was used for the processing of the structure of the masonic temple to capture the top part of the building. To generate a more homogeneous textured layer from multiple images, the images were white balanced using the diagonal model of illumination change. The Hennepin Avenue in Figure 5.4 was modeled in about one hour. The Masonic Temple is the most difficult one to model and it took about 10 minutes including re-modeling.

For the rendering results in the video, we assigned different reflective properties for

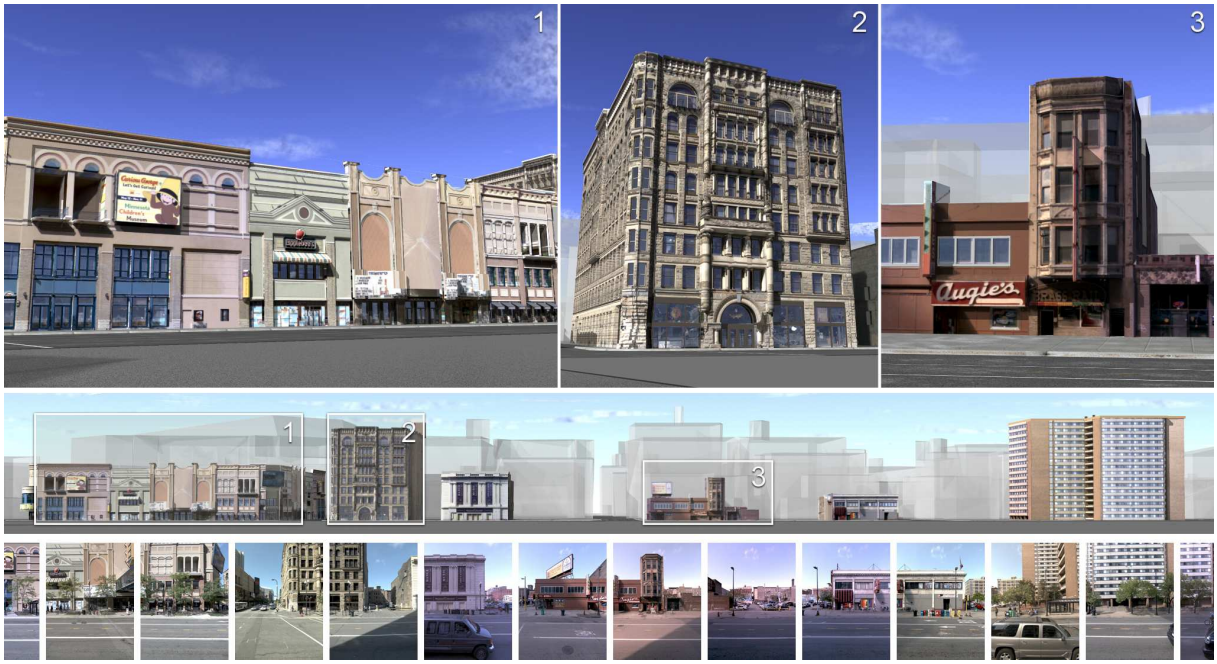


Figure 5.4: **Modeling of the Hennepin avenue in Minneapolis from 281 images.** Some input images are in the bottom row, the recovered model rendered is in the middle row, and three zoomed sections of the recovered model rendered are in the top row. This data set is provided by Microsoft Virtual Earth.

the windows and manually modeled the ground and vegetation. Our approach has been found to be efficient: Most manual post-editing was needed for visually important details near the roof tops of the buildings, where the common coverage of the images is small, and the quality of the recovered point cloud is poor. The re-modeling with generic models for clusters of patches is done only on the Hennepin Avenue example. It is obvious that the accuracy of the camera geometry and the density of reconstructed points are keys to the modeling. GPS/INS data did help to improve the registration of long sequences and avoid the drift associated with the SFM.

Typical façades Some typical façade examples from each data set are shown in Figure 5.1. An example from the Minneapolis data is also in the flowchart. We show both the before and after editing of the automatic partition, which shows that the majority of the façade partitions can be automatically computed with a over-segmentation followed by minor user adjustments. On average, the automatic computation time is about one minute per façade, then followed by about another minute of manual refinement.depending on the complexity and the desired reconstruction quality.

Difficult façade The masonic temple façade in the third row of Figure 5.1 shows the most difficult case that we encountered, mainly due to the specific capturing conditions. The overlapping of the images for the upper part of the building is small and we reconstruct only few points. The depth map for the upper part is almost constant after optimization. User interaction is more intensive to re-assign the depth for this façade.

Atypical façades Figure 5.5 shows some special façade examples that are also nicely handled by our approach.

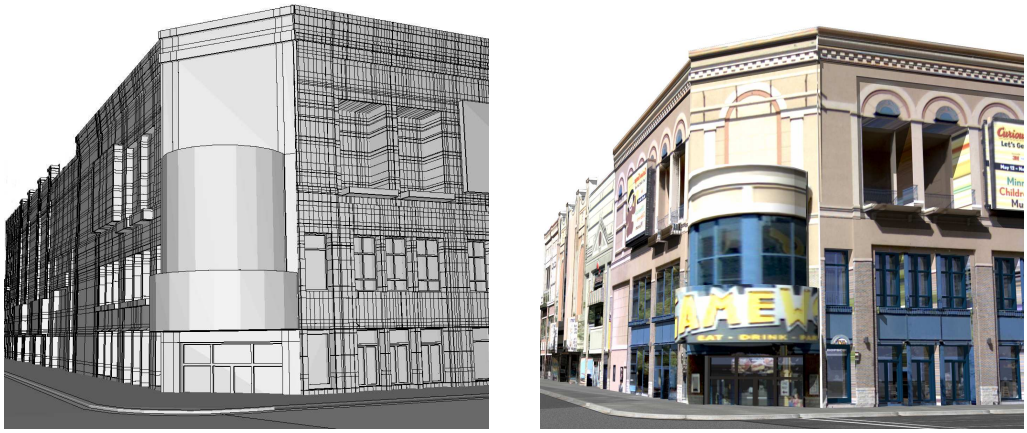
- **Cylindrical Façade** An example of cylindrical façade is illustrated in Figure 5.5(b). The cylindrical façade is modeled first, and then the second façade touched on it is modeled.
- **Re-modeling** This option was tested in the example of Hennepin avenue in Figure 5.4. The re-modeling results with 12 clusters shown in the top middle of Figure 5.5(c) can be compared with the results obtained without re-modeling shown on the right of Figure 5.1.
- **Multiple Façades** For the topologically complex building façades, we could use multiple façades. The whole Canton arcade street in Figure 5.3 systematically used two façade, the second façade uses the first façade in front as the occluders, shown in Figure 5.5(a).

We have presented an image-based street-side modeling approach that takes a sequence of overlapping images along the street, and produces complete photo-realistic 3D façade models. Our approach has several limitations for improvement as future work. The automatic depth reconstruction techniques may fail when trying to model highly reflective mirror-like buildings. And the reflectance properties of the models might be automatically recovered from multiple views. Furthermore, non-rectilinear objects might also be automatically detected during partition.

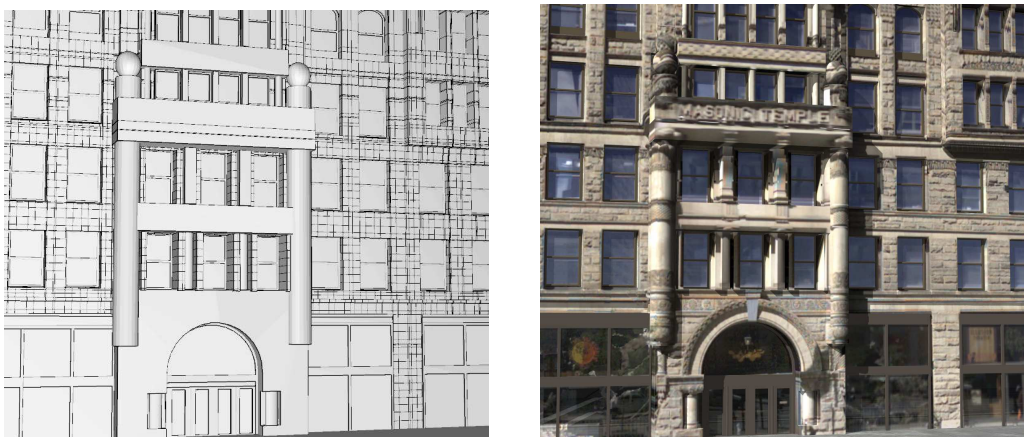
5.2 Automatic modeling

We have implemented our system and tested on the street-side images of downtown Pittsburgh. These images have been used in Google Street View to create seamless panoramic views. Therefore, the same kind of images is currently available for about 200 major cities in the world, which have been captured without online human control and with noises and glares. The image resolution is 640×905. Some example images are shown in Figure 5.8.

(a) Two Layers



(b) A cylindrical façade example



(c) Re-modeling by replacing a cube by a cylinder or a sphere

Figure 5.5: **Atypical façades examples.**

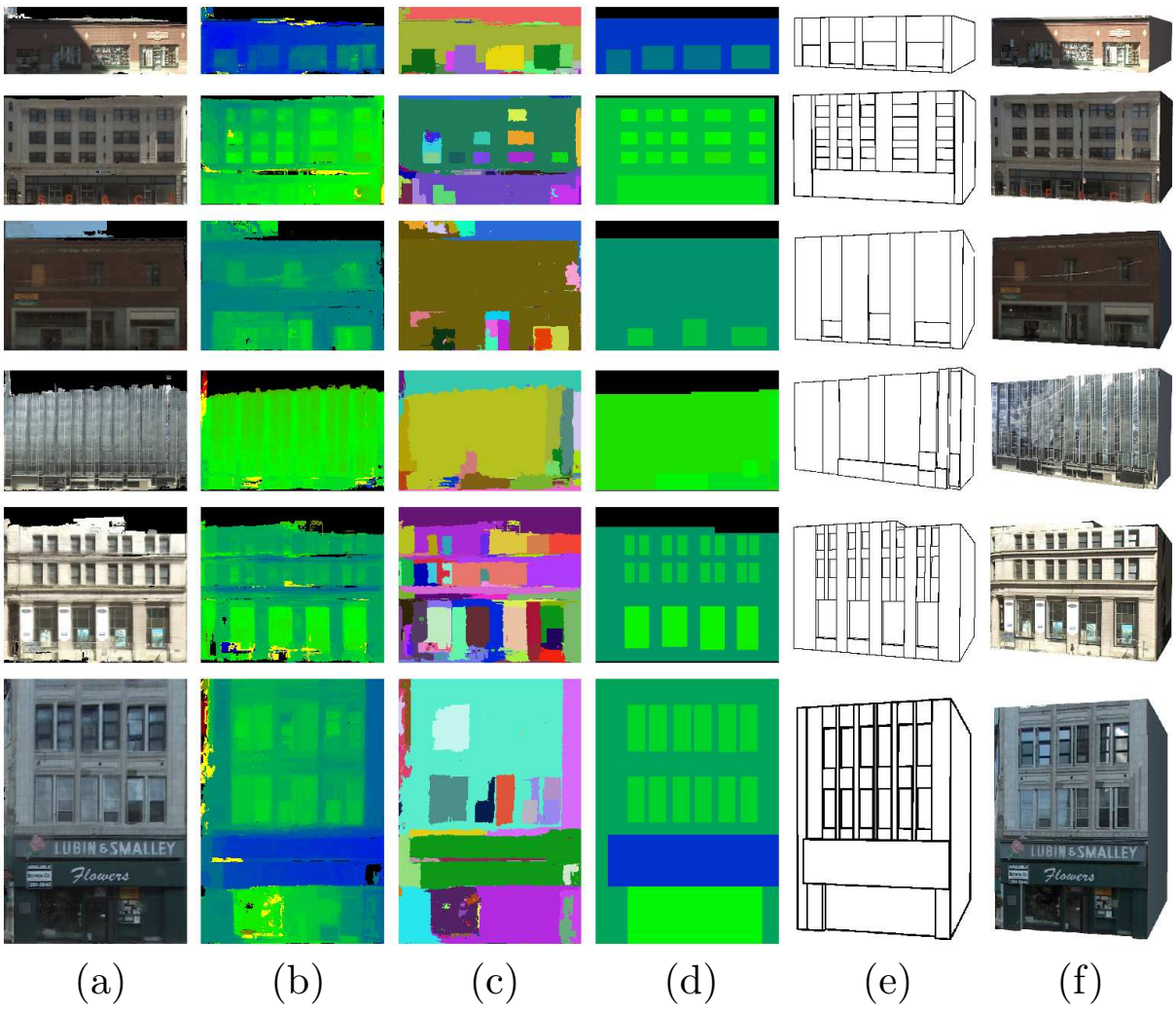


Figure 5.6: **Modeling examples of various blocks.** (a) The orthographic texture. (b) The orthographic color-coded depth map (yellow pixel is unreliable). (c) The façade segmentation. (d) The regularized depth map. (e) The geometry. (f) The textured model.

Each sequence is reconstructed using the structure from motion algorithm to produce a set of semi-dense points and camera poses. The cameras are then geo-registered back to the GPS coordinate frame. All sequences of a scene are merged with the overlapping camera poses.

We have implemented our methods with unoptimized C++ code, and tune the parameters manually on a set of 5 façades. For a portion of Pittsburgh, we used 10,498 images, and reconstructed 202 building blocks. On a small cluster composed by 15 normal desktop PCs, the results are produced automatically in 23 hours, including approximately 2 hours for SFM, 19 hours for segmentation, and 2 hours for partition and modeling. Figure 5.7 shows different examples of blocks and the intermediate results. Figure 5.9 shows a few close-up views of the final model. For rendering, each building block is represented in two levels of detail. The first level has only the façade base plane. The second level contains

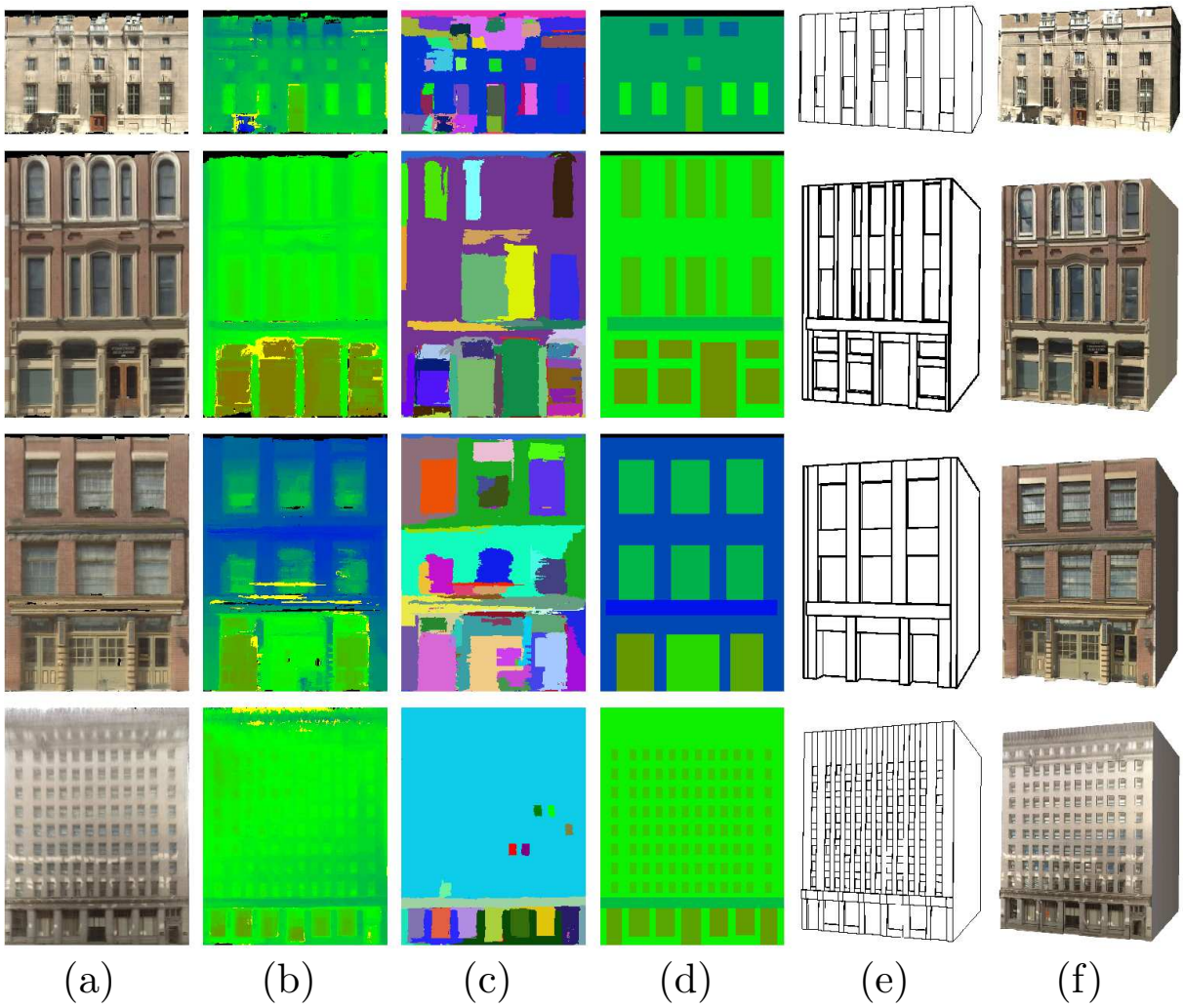


Figure 5.7: **Modeling examples of various blocks (Con't).** (a) The orthographic texture. (b) The orthographic color-coded depth map (yellow pixel is unreliable). (c) The façade segmentation. (d) The regularized depth map. (e) The geometry. (f) The textured model.

the augmented elements of the façade.

In the semantic segmentation, we hand-labeled 173 images by uniformly sampling images from our data set to create the initial database of labeled street-side images. Some example labeled data is shown in the accompanying video. Each sequence is recognized and segmented independently. For testing, we do not use any labeled images if they come from the same sequence in order to fairly demonstrate the real performance on unseen sequences.

Our method is remarkably robust for modeling as the minor errors or failure cases do not create visually disturbing artifacts. The distinct elements such as windows and doors within the façade may not always be reconstructed due to lack of reliable 3D points. They are often smoothed to the façade base plane with satisfactory textures as the depth

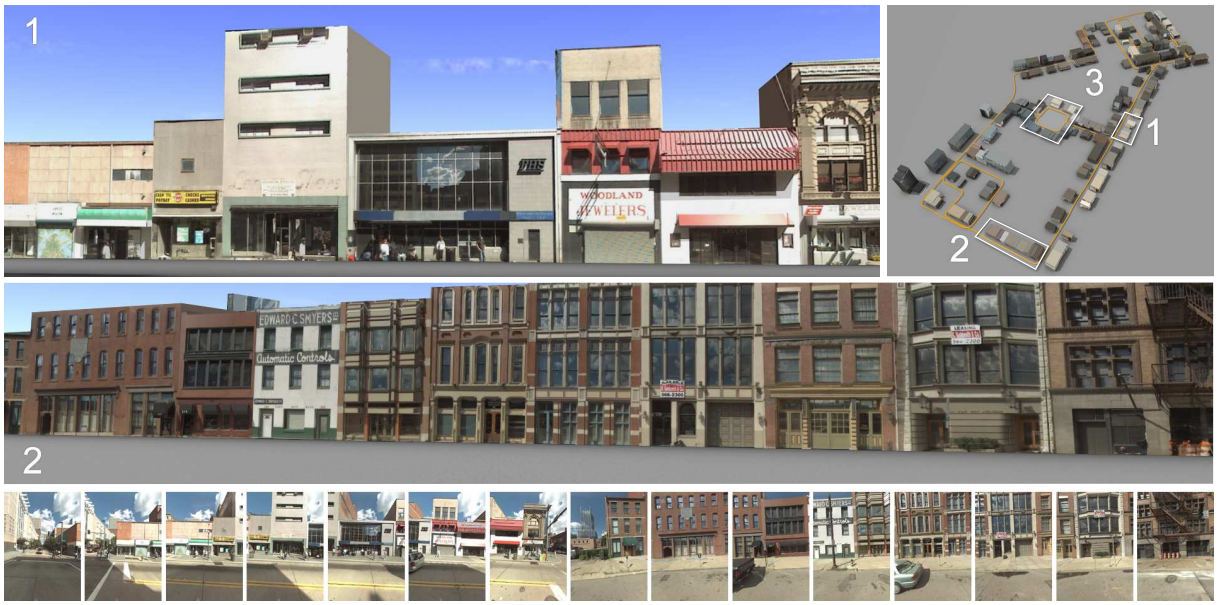


Figure 5.8: **Two close-up street-side views 1 and 2 of a modeled city area shown in the first two rows.** All the models are automatically generated from input images, exemplified by the bottom row. The close-up street-side view 3 is shown in Figure 5.9.

variation is small. Most of the artifacts are from the texture. Many of the trees and people are not removed from the textures on the first floor of the buildings seen in Figure 5.9. These could be corrected if an interactive segmentation and inpainting is used. There are some artifacts on the façade boundaries if the background buildings are not separated from the foreground buildings, shown in the middle of Figure 5.9. Some other modeling examples are also shown in Figure 5.7.

Limitations The restriction to the rectangular shape and trade-off for robustness is a limitation for more demanding modeling tasks such as landmark buildings. But the rectangular element can always be considered as the first level approximation. Then, it could be easily replaced by other objects or refined by other methods. With the limited viewing field of a single camera, we miss the upper parts of tall buildings. We can either merge the current street-side modeling of the lower parts of the buildings with the modeling results from aerial images for the upper parts, or we could deploy a multi-camera system with one of them pointing upward to capture the upper parts of the buildings. These limitations reflect more on the current implementation, but not on the framework that could be extended to remove these limitations.

Figure 5.9: **Two close-up street-side views of the city models automatically generated from the images shown on the bottom.**

(a) (b) (c) (d) (e) (f)

Figure 5.10: **Automatic reconstruction for Guangzhou dataset.** Our results are presented from (a) to (f) with the same legend as in Figure 5.7.

CHAPTER 6

CONCLUSION

In this thesis, we focus on a systematic computation pipeline for image-based building modeling, which takes a sequence of overlapping images captured along the street and produces the complete photo-realistic 3D models. Although this challenging problem has been studied for a long time in both academic research and commercial industrial communities, all the existing approaches still need plenty of human efforts in order to produce satisfactory results.

This thesis presents an automatic approach that only requires minimal human efforts for image-based building modeling to achieve visual pleasing results. This image-based approach has three steps: reconstruction, segmentation and modeling. Our three-step approach is remarkably robust, because it clearly divides the work into subproblems properly, and conquers each subproblem with strategies according to different objectives to be achieved in each stage. While this approach is also suitable for general image-based modeling of any object, specifical focus is on man-made buildings, where Manhattan-world property presents frequently.

The approach has been successfully demonstrated on large amount of data for building modeling in several cities, including Pittsburgh, Minneapolis, Chapel Hill in the United States, and Guangzhou in China. The proposed method is able to greatly improve the productivity for large-scale city modeling. Although there are a few limitations to the current implementation of the system, they can be improved within the same framework. As the reviewers of ACM Transaction on Graphics said, "Although the system still has limitations and the models show some artifacts, it is a clear step ahead for the state of the art" and "represents a significant progress beyond the state of the art."

REFERENCES

[1] E. Tola, V. Lepetit, and P. Fua, "A fast local descriptor for dense matching," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[2] C. Strecha, R. Fransens, and L. Van Gool, "Combined depth and outlier estimation in multi-view stereo," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, pp. 2394–2401.

[3] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, "A comparison and evaluation of multi-view stereo reconstruction algorithms," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, June 2006, pp. 519–526.

[4] J. Xiao, J. Chen, D.-Y. Yeung, and L. Quan, "Structuring visual words in 3d for arbitrary-view object localization," in *Proceedings of European Conference on Computer Vision*, 2008.

[5] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision*, vol. 47, no. 1-3, pp. 7–42, 2002.

[6] P. Felzenszwalb and D. Huttenlocher, "Efficient belief propagation for early vision," *International Journal of Computer Vision*, vol. 70, 2004.

[7] Q. Yang, L. Wang, R. Yang, S. Wang, M. Liao, and D. Nistér, "Real-time global stereo matching using hierarchical belief propagation," in *Proceedings of The British Machine Vision Conference*, 2006, pp. 989–998.

[8] D. Dueck and B. J. Frey, "Affinity propagation webpage," http://www.psi.toronto.edu/affinitypropagation/.

[9] L. Quan, P. Tan, G. Zeng, L. Yuan, J. Wang, and S. B. Kang, "Image-based Plant Modeling." *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 599–604, 2006.

[10] A. Akbarzadeh, J.-M. Frahm, P. Mordohai, B. Clipp, C. Engels, D. Gallup, P. Merrell, M. Phelps, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewéius, R. Yang, G. Welch, H. Towles, D. Nisté, and M. Pollefeys, "Towards urban 3D reconstruction from video," in *Proceedings of International Symposium on 3D Data Processing, Visualization, and Transmission*, 2006.

[11] R. Burtch, "History of photogrammetry," Center for Photogrammetric Training, Ferris State University, Tech. Rep., 2008.

[12] "Trailblazer – édouard-gaston deville," http://www.nrcan-rncan.gc.ca/com/deptmini/traipion/edouard-gastondeville-eng.php, 12 2008.

[13] J. Albertz, "A look back – 140 years of "photogrammetry" – some remarks on the history of photogrammetry," *Photogrammetric Engineering and Remote Sensing*, May 2007.

[14] C. McGlone, E. Mikhail, and J. Bethel, *Manual of Photogrammetry*, 5th ed. American Society for Photogrammetry and Remote Sensing, 2004.

[15] P. Debevec, C. Taylor, and J. Malik, "Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach," *SIGGRAPH*, pp. 11–20, 1996.

[16] P. E. Debevec, "Modeling and rendering architecture from photographs," Ph.D. dissertation, University of California at Berkeley, Computer Science Division, Berkeley CA, 1996.

[17] S. Coorg, N. Master, and S. Teller, "Acquisition of a large pose-mosaic dataset," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1998, p. 872.

[18] S. Coorg and S. Teller, "Extracting textured vertical facades from controlled close-range imagery," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, 1999, p. 1625.

[19] M. E. Antone and S. Teller, "Automatic recovery of relative camera rotations for urban scenes," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2000, p. 2282.

[20] S. Teller, M. Antone, Z. Bodnar, M. Bosse, S. Coorg, M. Jethwa, and N. Master, "Calibrated, registered images of an extended urban area," *International Journal of Computer Vision*, vol. 53, no. 1, pp. 93–107, 2003.

[21] M. Antone and S. Teller, "Scalable, absolute position recovery for omni-directional image networks," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1. Los Alamitos, CA, USA: IEEE Computer Society, 2001, p. 398.

[22] P. Sand, "Long-range video motion estimation using point trajectories," Ph.D. dissertation, Cambridge, MA, USA, 2006, adviser-Teller, Seth.

[23] S. Coorg and S. Teller, "Spherical mosaics with quaternions and dense correlation," *International Journal of Computer Vision*, vol. 37, no. 3, pp. 259–273, 2000.

[24] C. Frueh, S. Jain, and A. Zakhor, "Data processing algorithms for generating textured 3d building facade meshes from laser scans and camera images," *International Journal of Computer Vision*, vol. 61, no. 2, pp. 159–184, 2005.

[25] C. Früh and A. Zakhor, "An automated method for large-scale, ground-based city model acquisition," *International Journal of Computer Vision*, vol. 60, no. 1, pp. 5–24, 2004.

[26] M. Ding, K. Lyngbaek, and A. Zakhor, "Automatic registration of aerial imagery with untextured 3d lidar models," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 0, 2008, pp. 1–8.

[27] C. Früh and A. Zakhor, "Constructing 3d city models by merging aerial and ground views," *IEEE Comput. Graph. Appl.*, vol. 23, no. 6, pp. 52–61, 2003.

[28] ——, "3d model generation for cities using aerial photographs and ground level laser scans," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2. Los Alamitos, CA, USA: IEEE Computer Society, 2001, p. 31.

[29] C. Frueh and A. Zakhor, "Automated reconstruction of building facades for virtual walk-thrus," in *SIGGRAPH '03: ACM SIGGRAPH 2003 Sketches & Applications*. New York, NY, USA: ACM, 2003, pp. 1–1.

[30] C. Frueh, R. Sammon, and A. Zakhor, "Automated texture mapping of 3d city models with oblique aerial imagery," in *Proceedings of International Symposium on 3D Data Processing, Visualization, and Transmission*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 396–403.

[31] C. Frueh, S. Jain, and A. Zakhor, "Data processing algorithms for generating textured 3d building facade meshes from laser scans and camera images," *International Journal of Computer Vision*, vol. 61, no. 2, pp. 159–184, 2005.

[32] T. Pintaric, A. Rizzo, and U. Neumann, "Video-based virtual environments," in *SIGGRAPH '03: ACM SIGGRAPH 2003 Sketches & Applications*. New York, NY, USA: ACM, 2003, pp. 1–1.

[33] J. Hu, S. You, and U. Neumann, "Integrating lidar, aerial image and ground images for complete urban building modeling," in *Proceedings of International Symposium on 3D Data Processing, Visualization, and Transmission*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 184–191.

[34] L. Wang, S. You, and U. Neumann, "Large-scale urban modeling by combining ground level panoramic and aerial imagery," in *Proceedings of International Symposium on 3D Data Processing, Visualization, and Transmission*. IEEE Computer Society, 2006, pp. 806–813.

[35] J. Hu, S. You, and U. Neumann, "Vanishing hull," in *Proceedings of International Symposium on 3D Data Processing, Visualization, and Transmission*. IEEE Computer Society, 2006, pp. 448–455.

[36] L. Wang and U. Neumann, "A robust approach for automatic registration of aerial images with untextured aerial lidar data," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2009.

[37] L. Wang, U. Neumann, and S. You, "Wide-baseline image matching using line signatures," in *Proceedings of IEEE Computer Society International Conference on Computer Vision*, 2009.

[38] L. Wang, S. You, and U. Neumann, "Semiautomatic registration between ground-level panoramas and an orthorectified aerial image for building modeling," in *Proceedings of VRML(Virtual Representation and Modeling of Large-Scale Environments) workshop in ICCV07*, 2007.

[39] O. Wang, S. K. Lodha, and D. P. Helmbold, "A bayesian approach to building footprint extraction from aerial lidar data," in *Proceedings of International Symposium on 3D Data Processing, Visualization, and Transmission*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 192–199.

[40] S. K. Lodha, E. J. Kreps, D. P. Helmbold, and D. Fitzpatrick, "Aerial lidar data classification using support vector machines (svm)," in *Proceedings of International Symposium on 3D Data Processing, Visualization, and Transmission*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 567–574.

[41] M. Pollefeys, D. Nistér, J.-M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S.-J. Kim, P. Merrell, C. Salmi, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewénius, R. Yang, G. Welch, and H. Towles, "Detailed real-time

urban 3d reconstruction from video," *International Journal of Comptuer Vision*, 2007.

[42] V.Vezhnevets, A.Konushin, and A.Ignatenko, "Interactive image-based urban modeling," in *PIA-2007*, Munich, Germany, 2007, pp. 63–68.

[43] O. Barinova, A. Kuzmishkina, A. Vezhnevets, and V. Vezhnevets, "Learning class specific edges for vanishing point estimation," in *Proceedings of Graphicon*, Moscow, Russia, 2007.

[44] V.Konushin and V.Vezhnevets, "Automatic building texture completion," in *Proceedings of Graphicon*, Moscow, Russia, 2007, pp. 174–177.

[45] A. Yakubenko and A. Konushin, "Image-based 3d reconstruction of buildings," in *Proceedings of 14th Conference of students and young scientists "Lomonosov"*, Moscow, Russia, 2007.

[46] G. Krivovyaz and A. Yakubenko, "Raster cities 2d maps analysis for 3d urban modeling," in *Proceedings of 14th Conference of students and young scientists "Lomonosov"*, Moscow, Russia, 2007.

[47] A. Bogdanov and A. Yakubenko, "3d reconstruction of urban buildings from a single image via automatic image rectification," in *Proceedings of 15th Conference of students and young scientists "Lomonosov"*, Moscow, Russia, 2008.

[48] K. V. and K. V., "Facade texture reconstruction covered with wires," in *Proceedings of 15th Conference of students and young scientists "Lomonosov"*, Moscow, Russia, 2008.

[49] I. Stamos and P. K. Allen, "Geometry and texture recovery of scenes of large scale," *Computer Vision and Image Understanding*, vol. 88, no. 2, pp. 94–118, 2002.

[50] I. Stamos and M. Leordeanu, "Automated feature-based range registration of urban scenes of large scale," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2. Los Alamitos, CA, USA: IEEE Computer Society, 2003, p. 555.

[51] ——, "Efficient model creation of large structures based on range segmentation," in *Proceedings of International Symposium on 3D Data Processing, Visualization and Transmission*, September 2004, pp. 447–454.

[52] L. Liu and I. Stamos, "Automatic 3D to 2D registration for the photorealistic rendering of urban scenes," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005, pp. 137–143.

[53] I. Stamos, G. Yu, G. Wolberg, and S. Zokai, "3D modeling using planar segments and mesh elements," in *Proceedings of International Symposium on 3D Data Processing, Visualization, and Transmission*, 2006, pp. 599–606.

[54] L. Liu and I. Stamos, "A systematic approach for 2d-image to 3d-range registration in urban environments," in *Proceedings of IEEE Computer Society International Conference on Computer Vision*, Oct. 2007, pp. 1–8.

[55] I. Stamos, L. Liu, C. Chen, G. Wolberg, G. Yu, and S. Zokai, "Integrating automated range registration with multiview geometry for the photorealistic modeling of large-scale scenes," *International Journal of Computer Vision*, vol. 78, no. 2-3, pp. 237–260, 2008.

[56] A. Criminisi, I. D. Reid, and A. Zisserman, "Single view metrology," *International Journal of Computer Vision*, vol. 40, no. 2, pp. 123–148, 2000.

[57] B. M. Oh, M. Chen, J. Dorsey, and F. Durand, "Image-based modeling and photo editing," *SIGGRAPH*, pp. 433–442, 2001.

[58] D. Hoiem, A. A. Efros, and M. Hebert, "Automatic photo pop-up," in *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, 2005, pp. 577–584.

[59] A. Saxena, M. Sun, and A. Y. Ng, "Make3d: Learning 3d scene structure from a single still image," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 824–840, 2009.

[60] A. Torralba and A. Oliva, "Depth estimation from image structure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 9, pp. 1226–1238, 2002.

[61] E. B. Sudderth, A. Torralba, W. T. Freeman, and A. S. Willsky, "Depth from familiar objects: A hierarchical model for 3d scenes," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, pp. 2410–2417.

[62] O. Barinova, V. Konushin, A. Yakubenko, K. Lee, H. Lim, and A. Konushin, "Fast automatic single-view 3-d reconstruction of urban scenes," in *Proceedings of European Conference on Computer Vision*, 2008.

[63] A. van den Henge, A. Dick, T. Thormählen, B. Ward, and P. H. S. Torr, "Videotrace: rapid interactive scene modelling from video," *ACM Transactions on Graphics (SIGGRAPH)*, vol. 3, p. 86, 2007.

[64] J. Xiao, T. Fang, P. Tan, P. Zhao, E. Ofek, and L. Quan, "Image-based façade modeling," *ACM Transactions on Graphics*, vol. 27, no. 5, pp. 1–10, 2008.

[65] S. N. Sinha, D. Steedly, R. Szeliski, M. Agrawala, and M. Pollefeys, "Interactive 3D architectural modeling from unordered photo collections," *ACM Transactions on Graphics*, vol. 27, no. 5, pp. 1–10, 2008.

[66] T. Werner and A. Zisserman, "Model selection for automated architectural reconstruction from multiple views," in *Proceedings of the British Machine Vision Conference*, 2002, pp. 53–62.

[67] G. Schindler, P. Krishnamurthy, and F. Dellaert, "Line-based structure from motion for urban environments," in *Proceedings of International Symposium on 3D Data Processing, Visualization, and Transmission*, 2006, pp. 846–853.

[68] A. Dick, P. Torr, and R. Cipolla, "Modelling and interpretation of architecture from several images," *International Journal of Computer Vision*, vol. 2, pp. 111–134, 2004.

[69] K. Schindler and J. Bauer, "A model-based method for building reconstruction," *Higher-Level Knowledge in 3D Modeling and Motion Analysis, IEEE International Workshop on*, vol. 0, p. 74, 2003.

[70] ——, "Towards feature-based building reconstruction from images," in *Proceedings of WSCG*, 2003.

[71] ——, "Detailed building reconstruction with shape templates," in *Proceedings of 27th Workshop of the Austrian Association for Pattern Recognition (OeAGM)*, 2003.

[72] N. Cornelis, B. Leibe, K. Cornelis, and L. V. Gool, "3D urban scene modeling integrating recognition and reconstruction," *International Journal of Computer Vision*, 2007.

[73] N. Cornelis, K. Cornelis, and L. Van Gool, "Fast compact city modeling for navigation pre-visualization," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, pp. 1339–1344.

[74] N. Cornelis, B. Leibe, K. Cornelis, and L. Van Gool, "3D city modeling using cognitive loops," in *Proceedings of International Symposium on 3D Data Processing, Visualization, and Transmission*, 2006, pp. 9–16.

[75] Z. Lukas, K. Andreas, G.-G. Barbara, and K. Konrad, "Towards 3d map generation from digital aerial images," *International Journal of Photogrammetry and Remote Sensing*, vol. 60, pp. 413–427, 2006.

[76] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: Exploring photo collections in 3d," *ACM Transactions on Graphics (SIGGRAPH)*, vol. 3, pp. 835–846, 2006.

[77] ——, "Modeling the world from internet photo collections," *International Journal of Computer Vision*, 2007.

[78] M. Goesele, N. Snavely, B. Curless, S. M. Seitz, and H. Hoppe, "Multi-view stereo for community photo collections," in *Proceedings of IEEE Computer Society International Conference on Computer Vision*, 2007.

[79] N. Snavely, R. Garg, S. M. Seitz, and R. Szeliski, "Finding paths through the world's photos," *ACM Transactions on Graphics (SIGGRAPH)*, vol. 27, no. 3, pp. 11–21, 2008.

[80] B. Micusik and J. Kosecka, "Piecewise planar city 3d modeling from street view panoramic sequences," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2009.

[81] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski, "Manhattan-world stereo," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2009.

[82] ——, "Reconstructing building interiors from images," in *Proceedings of IEEE Computer Society International Conference on Computer Vision*, 2009.

[83] D. Gallup, J.-M. Frahm, P. Mordohai, Q. Yang, and M. Pollefeys, "Real-time plane-sweeping stereo with multiple sweeping directions," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

[84] F. Han and S. Zhu, "Bottom-up/top-down image parsing by attribute graph grammar," in *Proceedings of IEEE Computer Society International Conference on Computer Vision*, 2005, pp. 1778–1785.

[85] A. C. Berg, F. Grabler, and J. Malik, "Parsing images of architectural scenes," in *Proceedings of IEEE Computer Society International Conference on Computer Vision*, vol. 0. Los Alamitos, CA, USA: IEEE Computer Society, 2007, pp. 1–8.

[86] J. Košecká and W. Zhang, "Extraction, matching, and pose recovery based on dominant rectangular structures," *Computer Vision and Image Understanding*, vol. 100, no. 3, pp. 274–293, 2005.

[87] A. C. Murilo and J. Košecká, "Image based door recognition," in *IROS, From Sensors to Human Spatial Concepts*, 2007.

[88] W. Zhang and J. Košecká, "Hierarchical building recognition," *Image and Vision Computing*, vol. 25, no. 5, pp. 704–716, 2007.

[89] B. Micusik, H. Wildenauer, and J. Kosecka, "Detection and matching of rectilinear structures," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 2008, pp. 1–7.

[90] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*, 2nd ed. Cambridge University Press, 2004.

[91] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, "A comparison and evaluation of multi-view stereo reconstruction algorithms," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, 2006, pp. 519–528.

[92] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision*, vol. 47, no. 1-3, pp. 7–42, 2002.

[93] C. Strecha, R. Fransens, and L. Gool, "Wide-baseline stereo from multiple views: a probabilistic account," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, 2004, pp. 552–559.

[94] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[95] A. Toshev, J. Shi, and K. Daniilidis, "Image matching via saliency region correspondences," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[96] M. Lhuillier and L. Quan, "Match propagation for image-based modeling and rendering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 8, pp. 1140–1146, 2002.

[97] J. Kannala and S. S. Brandt, "Quasi-dense wide baseline matching using match propagation," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[98] S. Roth and M. J. Black, "Fields of experts: A framework for learning image priors," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2005, pp. 860–867.

[99] S. Roweis and L. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, pp. 2323–2326, 2000.

[100] J. Ham, D. Lee, and L. K. Saul, "Learning high dimensional correspondences from low dimensional manifolds," in *Proceedings of International Conference on Machine Learning*, 2003.

[101] J. Sun, Y. Li, S. B. Kang, and H. Y. Shum, "Symmetric stereo matching for occlusion handling," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2005, pp. 399–406.

[102] J. Krüger and R. Westermann, "Linear algebra operators for GPU implementation of numerical algorithms," *ACM Transactions on Graphics*, pp. 908–916, 2003.

[103] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Advances in Neural Information Processing Systems*, vol. 16, 2004, pp. 321–328.

[104] F. Wang, J. Wang, C. Zhang, and H. Shen, "Semi-supervised classification using linear neighborhood propagation," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, pp. 160–167.

[105] D. Hoiem, A. Stein, A. Efros, and M. Hebert, "Recovering occlusion boundaries from a single image," in *Proceedings of IEEE Computer Society International Conference on Computer Vision*, 2007.

[106] M. Lhuillier and L. Quan, "A quasi-dense approach to surface reconstruction from uncalibrated images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 3, pp. 418–433, 2005.

[107] L. Quan, "Invariant of six points and projective reconstruction from three uncalibrated images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 1, pp. 34–46, 1995.

[108] Weiss and Freeman, "On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs," *IEEETIT*, vol. 47, 2001.

[109] E. Mouragnon, F. Dekeyser, M. Lhuillier, M. Dhome, and P. Sayd, "Real time localization and 3d reconstruction," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, 0-0 2006, pp. 363–370.

[110] B. Frey and D. Dueck, "Mixture modeling by affinity propagation," in *Advances in Neural Information Processing Systems*, 2006, pp. 379–386.

[111] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, pp. 972–976, February 2007.

[112] D. Dueck and B. J. Frey, "Non-metric affinity propagation for unsupervised image categorization," in *Proceedings of IEEE Computer Society International Conference on Computer Vision*, 2007.

[113] M. Leone, Sumedha, and M. Weigt, "Clustering by soft-constraint affinity propagation: Applications to gene-expression data," *Bioinformatics*, vol. 2708, 2007.

[114] ——, "Semi-supervised clustering using soft-constraint affinity propagation," *Submitted to PRL (Available at http://arxiv.org/abs/0712.1165v1).*

[115] B. J. Frey and D. Dueck, "Response to comment on "clustering by passing messages between data point"," *Science*, vol. 319, no. 726d, Febrary 2008.

[116] M. J. Brusco and H.-F. Köhn, "Comment on "clustering by passing messages between data points"," *Science*, vol. 319, no. 5864, February 2008.

[117] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society*, vol. 39, no. 1, pp. 1–38, 1977.

[118] D. Lashkari and P. Golland, "Convex clustering with exemplar-based models," in *Advances in Neural Information Processing Systems*, 2008, pp. 825–832.

[119] L. Quan, J. Wang, P. Tan, and L. Yuan, "Image-based Modeling by Joint Segmentation." *International Journal of Computer Vision*, vol. To Appear, 2007.

[120] V. Kolmogorov, "Convergent Tree-Reweighted Message Passing for Energy Minimization," in *Proceedings of International Conference on Artificial Intelligence and Statistics*, 2005.

[121] X. Zhu and Z. Ghahramani, "Learning from Labeled and Unlabeled Data with Label Propagation," Tech. Rep. tech report CMU-CALD-02-107, 2002.

[122] M. Lhuillier and L. Quan, "A Quasi-Dense Approach to Surface Reconstruction from Uncalibrated Images." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 3, pp. 418–433, 2005.

[123] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[124] Z. Tu and S. C. Zhu, "Image Segmentation by Data-Driven Markov Chain Monte Carlo." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 657–673, 2002.

[125] P. Tan, G. Zeng, J. Wang, S. B. Kang, and L. Quan, "Image-based Tree Modeling." *ACM Transactions on Graphics*, 2007.

[126] J. Wang and E. Adelson, "Representing Moving Images with Layers." *IEEE Transactions on Image Processing*, vol. 3, no. 5, pp. 625–638, 1994.

[127] J. Wills, S. Agarwal, and S. Belongie, "What Went Where." in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2003, pp. 37–44.

[128] J. Xiao and M. Shah, "Motion Layer Extraction in the Presence of Occlusion Using Graph Cut." in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2004, pp. 972–979.

[129] I. Patras, E. Hendriks, and R. Lagendijk, "Video Segmentation by MAP Labeling of Watershed Segments." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 3, pp. 326–332, 2001.

[130] V. Kolmogorov and R. Zabih, "What energy functions can be minimized via graph cuts?" in *Proceedings of European Conference on Computer Vision*, 2002, pp. 65–81.

[131] V. Kolmogorov, A. Criminisi, A. Blake, G. Cross, and C. Rother, "Bi-Layer Segmentation of Binocular Stereo Video." in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005, pp. 407–414.

[132] J. Malik, S. Belongie, T. Leung, and J. Shi, "Contour and Texture Analysis for Image Segmentation." *International Journal of Computer Vision*, vol. 43, no. 1, pp. 7–27, 2001.

[133] E. Hadjidemetriou, M. Grossberg, and S. Nayar, "Multiresolution Histograms and Their Use for Texture Classification," in *Proceedings of 3rd International Workshop on Texture Analysis and Synthesis*, Oct 2003.

[134] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum, "Lazy Snapping." in *Proceedings of ACM SIGGRAPH*, 2004, pp. 303–308.

[135] Y. Schnitman, Y. Caspi, D. Cohen-Or, and D. Lischinski, "Inducing semantic segmentation from an example," in *Proceedings of Asian Conference on Computer Vision*, 2006, pp. 373–384.

[136] J. Xiao and L. Quan, "Multiple view semantic segmentation for street view images," in *Proceedings of IEEE Computer Society International Conference on Computer Vision*, 2009.

[137] P. Tan, T. Fang, J. Xiao, P. Zhao, and L. Quan, "Single image tree modeling," *ACM Transactions on Graphics*, vol. 27, no. 5, pp. 1–7, 2008.

[138] J. Corso, A. Yuille, and Z. Tu, "Graph-shifts: Natural image labeling by dynamic hierarchical computing," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.

[139] G. Csurka and F. Perronnin, "A simple high performance approach to semantic segmentation," in *Proceedings of The British Machine Vision Conference*, 2008.

[140] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie, "Objects in context," in *Proceedings of IEEE Computer Society International Conference on Computer Vision*, 2007.

[141] J. Shotton, J. Winn, C. Rother, and A. Criminisi, "Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context," *International Journal of Computer Vision*, vol. 81, no. 1, pp. 2–23, 1 2009.

[142] J. Shotton, M. Johnson, and R. Cipolla, "Semantic texton forests for image categorization and segmentation," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.

[143] G. Stephen, R. Jim, C. David, E. Gal, and K. Daphne, "Multi-class segmentation with relative location prior," *International Journal of Computer Vision*, vol. 80, no. 3, pp. 300–316, 2008.

[144] L. Zhu, Y. Chen, Y. Lin, and A. Yuille, "A hierarchical image model for polynomial-time 2d parsing," in *Advances in Neural Information Processing Systems*, 2008.

[145] E. B. Sudderth and M. I. Jordan, "Shared segmentation of natural scenes using dependent pitman-yor processes," in *Advances in Neural Information Processing Systems*, 2008.

[146] A. Thomas, V. Ferrari, B. Leibe, T. Turtelaars, B. Schiele, and L. V. Gool, "Towards multi-view object class detection," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006.

[147] S. Savarese and L. Fei-Fei, "3D generic object categorization, localization and pose estimation," in *Proceedings of IEEE Computer Society International Conference on Computer Vision*, 2007.

[148] J. Liebelt, C. Schmid, and K. Schertler, "Viewpoint-independent object class detection using 3D feature maps," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.

[149] A. P. IV, P. Mordohai, and K. Daniilidis, "Object detection from large-scale 3d datasets using bottom-up and top-down descriptors," in *Proceedings of European Conference on Computer Vision*, 2008.

[150] C. Christoudias, R. Urtasun, and T. Darrell, "Unsupervised feature selection via distributed coding for multi-view object recognition," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.

[151] J. Xiao, J. Wang, P. Tan, and L. Quan, "Joint affinity propagation for multiple view segmentation," in *Proceedings of IEEE Computer Society International Conference on Computer Vision*, 2007.

[152] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla, "Segmentation and recognition using structure from motion point clouds," in *Proceedings of European Conference on Computer Vision*, 2008.

[153] I. Posner, D. Schroeter, and P. Newman, "Describing composite urban workspaces," in *Proceedings of IEEE Computer Society International Conference on Robotics and Automation*, 2007.

[154] M. Lhuillier and L. Quan, "A quasi-dense approach to surface reconstruction from uncalibrated images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, pp. 418–433, 2005.

[155] J. Xiao, J. Chen, D.-Y. Yeung, and L. Quan, "Learning two-view stereo matching," in *Proceedings of European Conference on Computer Vision*, 2008.

[156] X. Ren and J. Malik, "Learning a classification model for segmentation," in *Proceedings of IEEE Computer Society International Conference on Computer Vision*, 2003.

[157] G. Mori, X. Ren, A. A. Efros, and J. Malik, "Recovering human body configurations: combining segmentation and recognition," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2004.

[158] S. Geman and D. Geman, "Stochastic relaxation, gibbs distributions, and the bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, no. 6, pp. 721–741, 1984.

[159] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 11, pp. 1222–1239, 2001.

[160] K. Barnard, P. Duygulu, D. Forsyth, N. de Freitas, D. M. Blei, and M. I. Jordan, "Matching words and pictures," *Journal of Machine Learning Research*, vol. 3, pp. 1107–1135, 2003.

[161] S. Gould, J. Rodgers, D. Cohen, G. Elidan, and D. Koller, "Multi-class segmentation with relative location prior," *International Journal of Computer Vision*, 2008.

[162] J. Winn, A. Criminisi, and T. Minka, "Object categorization by learned universal visual dictionary," in *Proceedings of IEEE Computer Society International Conference on Computer Vision*, 2005.

[163] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine Learning*, no. 3, pp. 297–336, 1999.

[164] S. K. Divvala, A. A. Efros, and M. Hebert, "Can similar scenes help surface layout estimation?" in *Proceedings of IEEE Workshop on Internet Vision at CVPR*, 2008.

[165] A. Oliva and A. Torralba, "Building the gist of a scene: The role of global image features in recognition," *Visual Perception, Progress in Brain Research*, vol. 155, 2006.

[166] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, pp. 972–976, 2007.

[167] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "Labelme: a database and web-based tool for image annotation," *International Journal of Computer Vision*, vol. 77, no. 1-3, pp. 157–173, 2008.

[168] L. von Ahn and L. Dabbish, "Labeling images with a computer game," in *Proceedings of ACM SIGCHI*, 2004, pp. 319–326.

[169] L. von Ahn, R. Liu, and M. Blum, "Peekaboom: a game for locating objects in images," in *Proceedings of ACM SIGCHI*, 2006, pp. 55–64.

[170] P. F. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008.

[171] D. Hoiem, A. A. Efros, and M. Hebert, "Putting objects in perspective," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006.

[172] B. Leibe, N. Cornelis, K. Cornelis, and L. V. Gool, "Dynamic 3d scene analysis from a moving vehicle," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 2007.

[173] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut: interactive foreground extraction using iterated graph cuts," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 309–314, 2004.

[174] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision*, vol. 59, no. 2, 9 2004.

[175] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 303–312.

[176] J. F. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, pp. 679–714, 1986.

[177] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, Jan 1972.

[178] P. Müller, G. Zeng, P. Wonka, and L. V. Gool, "Image-based procedural modeling of façades," *ACM Transactions on Graphics (SIGGRAPH)*, vol. 3, 2007.

[179] A. C. Berg, F. Grabler, and J. Malik, "Parsing images of architectural scenes," in *Proceedings of IEEE Computer Society International Conference on Computer Vision*, 2007.

[180] J. Pearl, "Reverend bayes on inference engines: A distributed hierarchical approach," in *Proceedings of AAAI Conference on Artificial Intelligence*, 1982, pp. 133–136.

[181] A. Criminisi, P. Perez, and K. Toyama, "Object removal by exemplar-based inpainting," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2003, pp. 721–728.

[182] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 313–318, 2003.

[183] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum, "Lazy snapping," *ACM Transactions on Graphics (SIGGRAPH)*, vol. 23, pp. 303–308, 2004.

[184] J. Sun, L. Yuan, J. Jia, and H.-Y. Shum, "Image completion with structure propagation," *ACM Transactions on Graphics*, vol. 24, pp. 861–868, 2005.