EFFICIENT 3D RECONSTRUCTION OF LARGE-SCALE URBAN ENVIRONMENTS FROM STREET-LEVEL VIDEO

David Gallup

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
2011

ABSTRACT

DAVID GALLUP: Efficient 3D Reconstruction of Large-Scale Urban Environments from Street-Level Video
(Under the direction of Marc Pollefeys and Jan-Michael Frahm)

Recovering the 3-dimensional (3D) structure of a scene from 2-dimensional (2D) images is a fundamental problem in computer vision. This technology has many applications in computer graphics, entertainment, robotics, transportation, manufacturing, security, etc. One application is 3D mapping. For example, Google Earth and Microsoft Bing Maps provide a 3D virtual replica of many of the Earth's cities. However, these 3D models are low-detail and lack ground-level realism. Google Street View and Bing Street Side provide high-resolution panoramas captured from the streets of many cities, but these stills cannot provide free navigation through the virtual world. In this dissertation, I will show how to automatically and efficiently create detailed 3D models of urban environments from street-level imagery.

A major goal of this dissertation is to model large urban areas, even entire cities, which is an enormous challenge due to the sheer scale of the problem. Even a partial data capture of the town of Chapel Hill requires millions of frames of street-level video. The methods presented in this dissertation are highly parallel and use little memory, and can therefore utilize modern graphics hardware (GPU) technology to process video at the recording frame rate. Also, the structure in urban scenes such as planarity, orthogonality, verticality, and texture regularity can be exploited to achieve 3D reconstructions with greater efficiency, higher quality, and lower complexity.

By examining the structure of an urban scene, a multiple-direction plane-sweep stereo method is performed on the GPU in real-time. An analysis of stereo precision leads to a view selection strategy that guarantees constant depth resolution and improves bounds on time complexity. Depth measurements are further improved by segmenting the scene into piecewise-planar and non-planar regions, a process which is aided by learned planar surface appearance. Finally, depth measurements are fused and the final 3D surface is recovered using a multi-layer heightmap model that produces clean, complete, and compact 3D reconstructions. The effectiveness of these methods is demonstrated by results from thousands of frames of video from a variety of urban scenes.

To Karen and Elijah

ACKNOWLEDGMENTS

I have received a tremendous amount of help and support from advisors, committee members, post-docs, fellow students, faculty, staff, and family members. Without their guidance and support, this dissertation would not have been possible.

I would like to thank my advisor, Marc Pollefeys, for seeing my potential and taking me on as a student. Many times when I felt I had reached a dead-end, Marc has been able to analyze the problem from a broader perspective and present new directions to persue. I am also grateful for my co-advisor, Jan-Michael Frahm, whose optimism and determination have allowed me to achieve more than I thought was possible. Jan's advice and suggestions have been invaluable.

I am fortunate to have had Greg Welch, Svetlana Lazebnik, and Carlo Tomasi on my committee. They are truly world-class researchers, and their feedback helped me gain new understanding about my own research.

The students and post-docs at the University of North Carolina have been fantastic. Thanks to Brian Clipp, Rahul Raguram, Changchang Wu, Paul Merrell, Seon Joo Kim, Sudipta Sinha, Li Guan, Ram Kumar, Tim Johnson, Yi-Hung Jen, Dibyendu Goswami, Christopher Zach, Philippos Mordohai, Hua Yang, Megha Pandey, and Joe Tighe. They have been inspirational collabrators, classmates, officemates, and friends.

I would like to thank the rest of the faculty and staff in the Department of Computer Science for providing me with an excellent education. I learned something valuable in each and every class, and I am impressed by the cheerful and hard-working attitude of everyone in the department.

I will never forget the time I spent at ETH Zurich. I would like to thank the students, post-docs, and staff who helped me with my research: Friedrich Fraundorfer, Gabriel Brostow, Luca Ballan, George Baatz, Roland Angst, Jens Puwein, Lorenz Meier, and Beatrice

Gander. I would also like to thank the students of the 3D Computer Vision course for their hard work and for making my teaching experience enjoyable.

Thanks to Cha Zhang, Zhengyou Zhang, and Qin Cai. My internship at Microsoft Research was fruitful in both research and career development.

I am grateful to my parents for their hard work and sacrifice to give me the gift of education. They encouraged me to believe I was smart enough to succeed in whichever career I chose, and they nurtured my curiosity about computers from an early age. Thanks for their enthusiastic support every step of the way. I would also like to thank my wife's parents who have been supportive of me delaying employment to take their daughter two thousand miles across the country for graduate school.

Finally I would like to thank my wife, Karen, who has worked every bit as hard as I have for my education. I could not have done it without you. And of course, I am grateful for my son, Elijah, who has been a joyful distraction during the difficult and stressful times.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

2D	Two Dimensional
3D	Three Dimensional
CCD	Charge-Coupled Device
CMOS	Complementary Metal-Oxide-Semiconductor
GPS	Global Positioning System
GPU	Graphics Processing Unit
HSV	Hue, Saturation, Value
INS	Inertial Navigation System
KLT	Kanade-Lucas-Tomasi (Feature Tracker)
LiDAR	Light Detection And Ranging
MRF	Markov Random Field
NCC	Normalized Cross-Correlation
PC	Personal Computer
RANSAC	Random Sample Consencus
RGB	Red, Green, Blue
SAD	Sum of Absolute Differences
SAR	Synthetic Aperture Radar
SfM	Structure from Motion
SIFT	Scale-Invariant Feature Transform
SSD	Sum of Squared Differences
UTM	Universal Transverse Mercator

CHAPTER 1

Introduction

Recovering the 3-dimensional (3D) structure of a scene from 2-dimensional (2D) images is a fundamental problem in computer vision. This technology has many applications in computer graphics, entertainment, robotics, transportation, manufacturing, security, etc. Furthermore, cameras and their images are becoming increasingly available. In addition to traditional video and still cameras, many mobile phones and automobiles now contain one or more cameras, and the Internet, especially websites such as Flickr [1] and Picasa [2], makes it easy to store and share these images. Computer vision provides the opportunity to enhance current applications with 3D information and create new uses for cameras and their images. One example pertinent to this dissertation is 3D digital mapping. Google Earth and Microsoft Bing Maps provide a 3D virtual replica of many of the Earth's cities. However, these 3D models are low-detail and lack ground-level realism. Google Street View and Bing Street Side provide high-resolution panoramas captured from the streets of many cities, but these stills cannot provide free navigation through the virtual world. Using 3D computer vision, specifically the methods presented in this dissertation, I will show how to automatically and efficiently create detailed 3D models of urban environments from street-level imagery. (See Figure 1.1).

Urban 3D modeling is an important domain that has received a lot of attention in the computer vision research community (Teller, 1998; Teller et al., 2003; Wang et al., 2002; Dick et al., 2001; Werner and Zisserman, 2002; Früh and Zakhor, 2003; Früh and Zakhor, 2004; Schindler and Dellaert, 2004; Schindler et al., 2006; Zebedin et al., 2006;

[1] http://www.flickr.com

[2] http://picasaweb.google.com

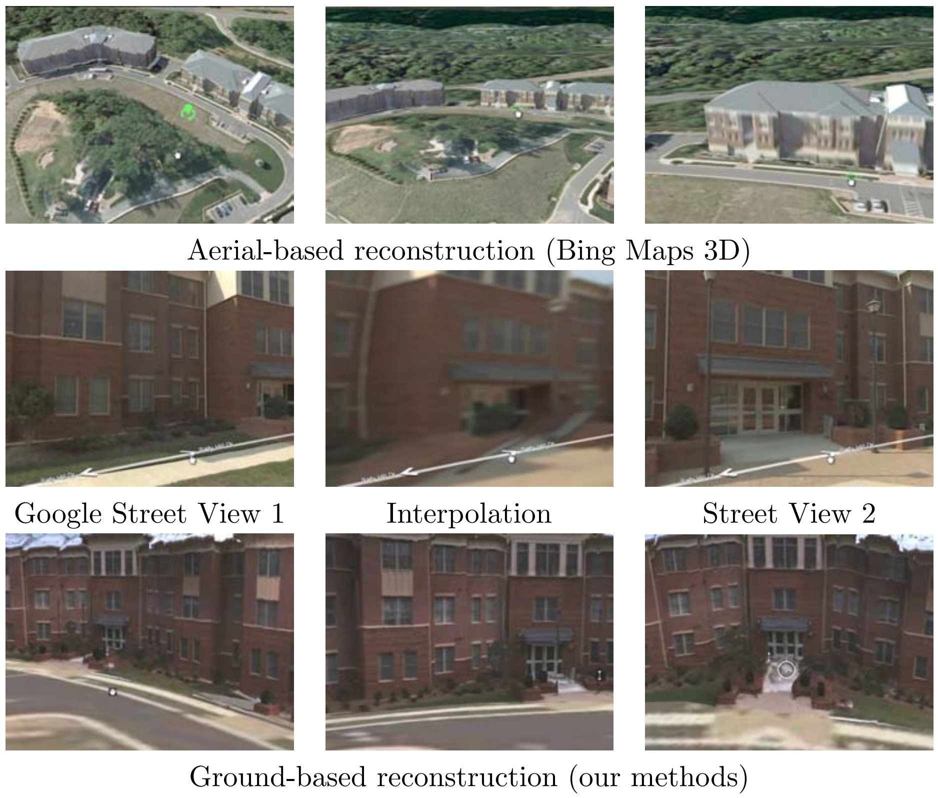

Aerial-based reconstruction (Bing Maps 3D)
Google Street View 1
Interpolation
Street View 2
Ground-based reconstruction (our methods)

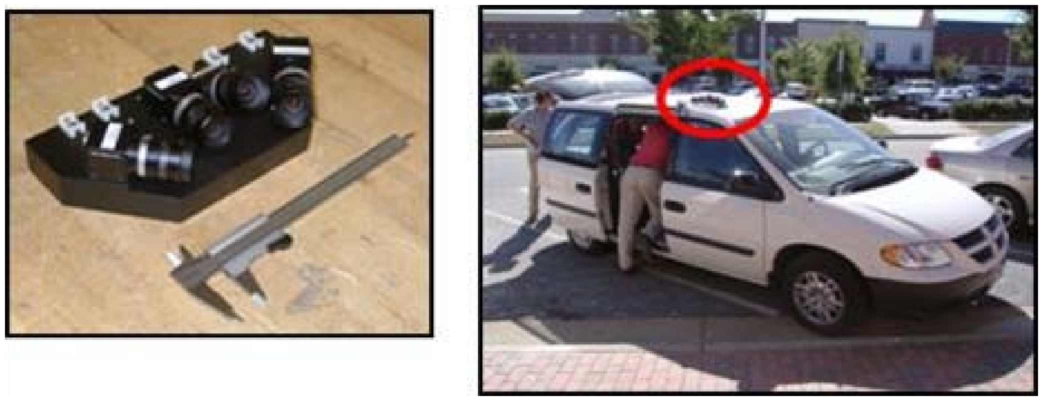
Figure 1.1: 3D reconstruction produces high-detail models for free-viewpoint navigation.

Zebedin et al., 2008; Furukawa et al., 2009b; Furukawa et al., 2009a) as well as in industry. Companies such as Google and Microsoft are investing much effort in acquiring digital maps of the Earth including photograph and video imagery from the street and air. This data can be used to reconstruct the world in 3D. Besides visualizing a 3D virtual world, these 3D models represent measurements which can be used for city planning, robot navigation, disaster response, and digital archiving.

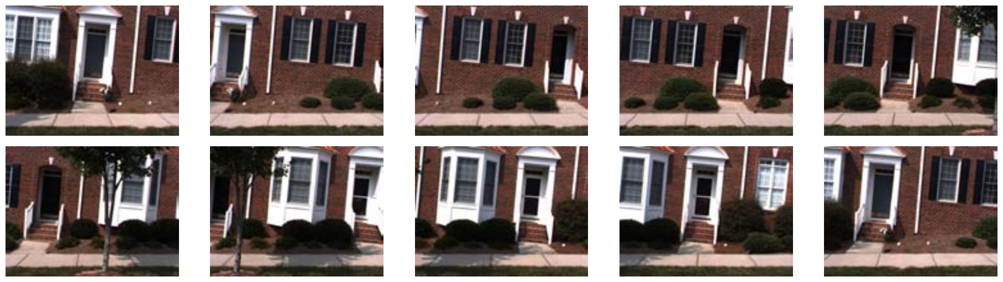
There are various techniques for 3D reconstruction including Light Detection And Ranging (LiDAR) (Früh and Zakhor, 2004), structured light (Scharstein and Szeliski, 2003), time-of-flight cameras (Guan et al., 2008; Schuon et al., 2008), and stereo (Scharstein and Szeliski, 2002). Many of these methods are not well suited for large-scale outdoor reconstruction. Structured light projectors are too weak relative to daylight, and time-of-flight cameras have only a short operating distance. LiDAR is a suitable technology, and successful city-scale reconstruction systems have been built for both aerial-based LiDAR (Morgan and Tempfli, 2000; You et al., 2003) and ground-based LiDAR (Früh and Zakhor, 2003; Früh and Zakhor, 2004). Stereo is now approaching the accuracy of LiDAR (Vu et al., 2009), and has several advantages. First, it is a passive system, using cameras that only capture the light energy naturally present in a scene. Compared to LiDAR, which must emit its own light signal, cameras require less power, are safer to operate, and can be used in clandestine operating scenarios (e.g. military). Also, LiDAR-based systems typically also have cameras to capture surface appearance for texture mapping.

Stereo can be used to process aerial or terrestrial imagery. Reconstruction from aerial and satellite imagery has long been studied in the photogrammetry community (Förstner, 1999; Baillard et al., 1999; Zebedin et al., 2006; Zebedin et al., 2008). However, many mapping applications such as virtual tourism need ground-level detail which cannot be captured from the air. Street-level video and photographs are a favorable solution, since streets are naturally accessible to vehicles and the recording equipment they carry. Capturing video or photographs from every street of a city can of course be expensive and time consuming. Ideally, ground-based and aerial-based methods should be viewed as complementary, with ground-based methods providing higher resolution, and aerial-based methods providing greater coverage, especially on rooftops or other areas invisible from the ground. Reconstructing indoor environments is another complementary problem which has received attention recently (Furukawa et al., 2009b; Saurer et al., 2010; Chen et al., 2010).

In this dissertation, I will present methods that automatically and efficiently reconstruct detailed 3D models of large-scale urban environments from video using stereo. These methods are an integral part of a larger system called *Urbanscape* (Pollefeys et al., 2008). The goal of the Urbanscape project is to capture street-level video from multiple automobile-mounted cameras, and to efficiently process that video to produce detailed 3D models. See Figures 1.2 and 1.3. The system uses structure from motion (SfM) as well as Global Positioning System (GPS) and inertial navigation system (INS) data to estimate the vehicle and camera pose through the recorded video sequence. The camera pose information and recorded video frames are then used to perform stereo to recover dense depth measurements of the surfaces visible in the video. Finally, the depth measurements are fused together to



Video cameras and example recording vehicle.



Some recorded video frames.

Figure 1.2: Urbanscape capture system.

produce a final surface estimate in the form of a texture-mapped triangle mesh. A major accomplishment of this project is that the system is able to process video in *real-time*, in other words *at the recording rate*.

My methods have contributed to the latter half of the Urbanscape processing pipeline, namely dense stereo, depthmap fusion, and surface modeling. These methods can handle most types of data, and results are shown from handheld video, calibrated still photographs, and even Internet photo collections. However, the main focus will be on reconstructing street-level video.

A major goal of this dissertation is to model large areas, even entire cities, which challenging due to the sheer scale of the problem. Even a partial data capture of the town of Chapel Hill requires 2.54 million of frames of video, equal to 1.97 terabytes of uncompressed frames. (See Figure 1.4.) In order to handle problems of this size, several factors must be considered:

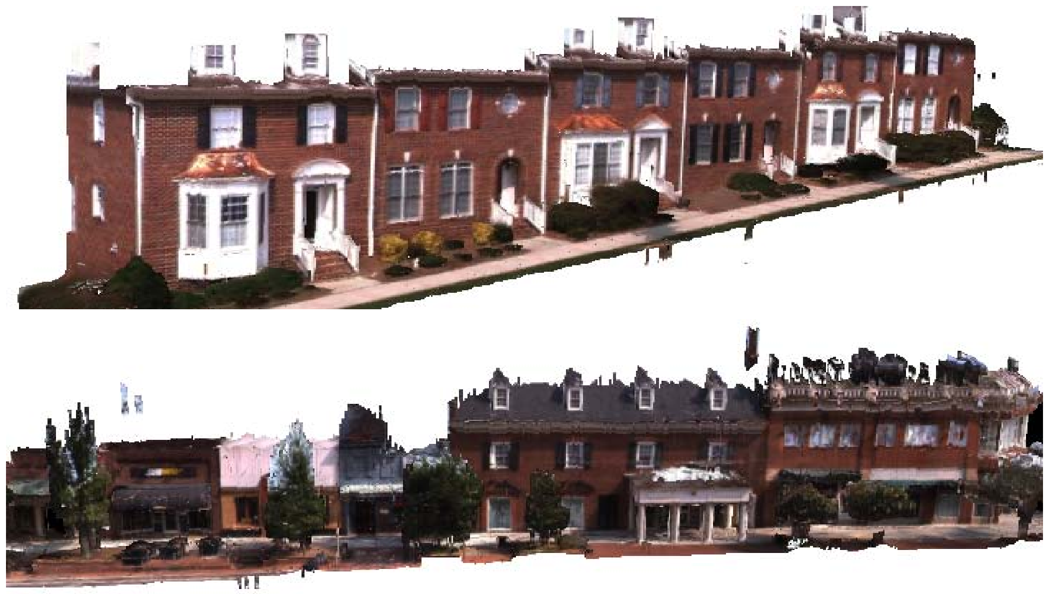


Figure 1.3: Urbanscape example 3D reconstructions.

Speed The algorithm must be fast enough to process large amounts of data in a reasonable amount of time. At the inception of the Urbanscape project, the state-of-the-art method of Pollefeys et al. (2004) could process video at about 1 minute per frame. At this rate, it would take a single PC 4.75 years to process 2.5 million frames. At present, the Urbanscape system, using the method discussed in this dissertation, can process the same data at 13.33 frames per second, taking less than 53 hours on a single PC. This is due in part to advances in CPU speed, but mostly due to the utilization of parallel algorithms which can be accelerated significantly on modern graphics hardware (GPU).

Memory Memory usage during processing is also a concern. Algorithms must exhibit locality, i.e. parts that can be processed independently, because such a large dataset cannot be stored in main memory. This *out-of-core* capability is one of the main features of the piecewise-planar reconstruction method presented in Chapter 6. Locality also enables distributed processing on a cluster of computers, which is another way to achieve parallelism besides GPUs.

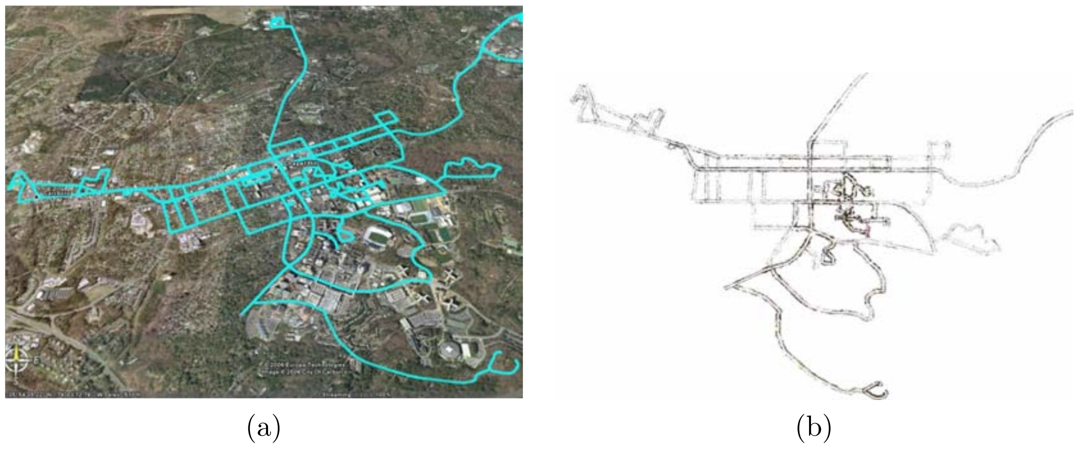


Figure 1.4: City-scale datasets require efficient, scalable processing. (a) The camera path of 2.54 million frames of street-level video. (b) The entire dataset processed in less than 1 day.

Storage Finally, long-term storage of the data, both input video and output models, is a significant challenge. The heightmap representation presented in Chapter 7 provides a compact way to store 3D urban reconstructions. Also note that if processing speed is faster than the recording rate, then in theory the input data does not need to be stored.

The key to successfully meeting these challenges is to take advantage of the unique and prevalent structure in urban scenes. (However, we will not require this structure to be present, and our methods can still process general scenes.) Figure 1.5 shows examples of the following features of urban environments:

- **planarity:** Planar surfaces like walls, rooftops, and sidewalks, and to some extent the ground, are prevalent in urban scenes.
- **verticality:** Walls are built vertically so as to support the weight of the structure under gravity. Of course there are exceptions.
- **orthogonality:** Walls often meet at right angles. The ground may also be orthogonal to walls, but in practice there is often some slope.
- **texture regularity:** Repeated surface structures such as brick, shingles, and siding have a regular appearance or texture.

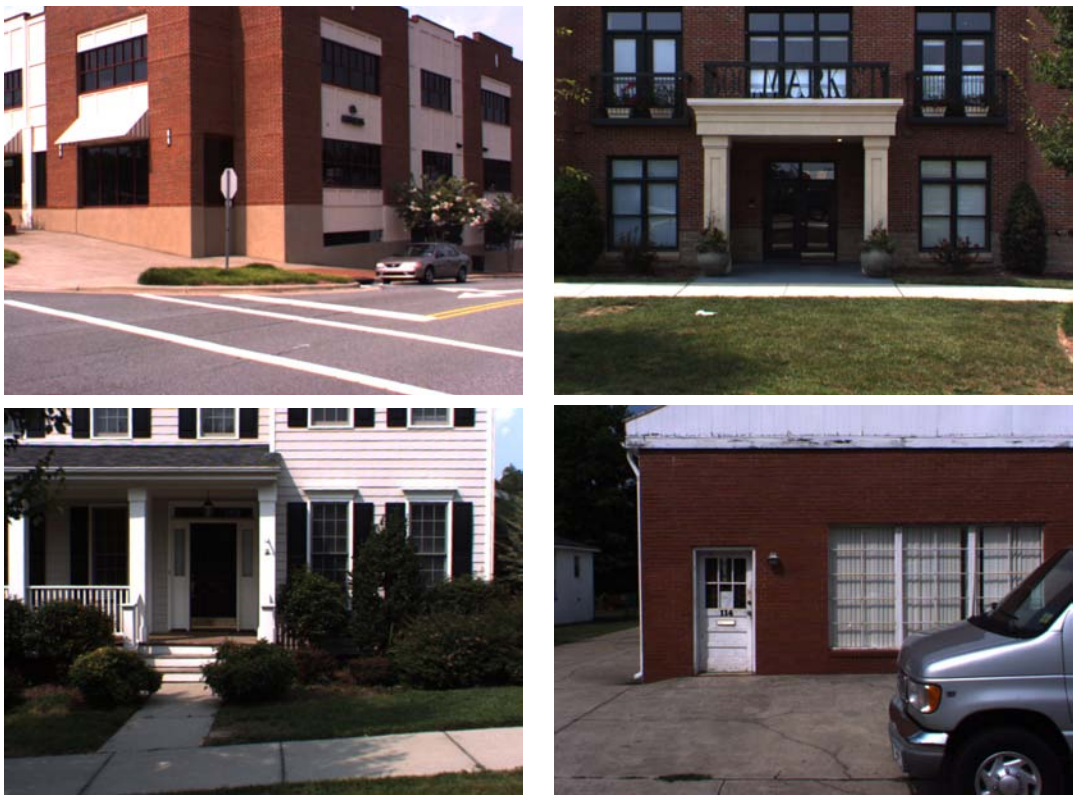


Figure 1.5: Urban scene structure includes planarity, orthogonality, verticality, and texture regularity.

- **color:** Color is a useful cue for distinguishing between man-made objects such as buildings and natural objects such as vegetation.

The methods presented in this dissertation aim to discover these types of structure in the scene, and use that information to achieve more accurate and more efficient reconstructions. Knowing the underlying structure, e.g. plane orientation or the vertical direction, the algorithm can achieve greater accuracy by biasing the solution towards the structure, and the algorithm can achieve greater efficiency by focusing on the discovered structure, avoiding the need to consider all general shapes. Nevertheless, the algorithms in this dissertation are capable of general shape reconstruction. One of the themes of this work is to enhance the reconstruction when urban structure is present while preserving the ability to handle the large variety of objects present in urban scenes.

1.1 Thesis Statement

Dense 3D reconstruction of large-scale urban environments can be performed automatically from video captured from street-level. By using parallelizable and scalable algorithms which take advantage of urban scene structure, the 3D scene can be reconstructed accurately and efficiently, even in real-time.

1.2 Contributions

My research makes the following contributions to urban 3D reconstruction:

Plane-sweeping stereo with multiple sweeping directions. (Chapter 4) I present a plane-sweep stereo method which runs in real-time on the GPU and uses multiple sweeps to achieve more accurate dense correspondence in urban scenes. The method consists of the following:

- The method analyzes the sparse point cloud resulting from structure-from-motion and computes the three principal surface normals of an urban scene: ground, facade, and side wall.
- A plane sweep in each principal direction allows the real-time stereo method to correctly handle slanted surfaces in the urban scene, and produces higher quality depthmaps.
- A plane position prior constructed from the sparse point cloud improves accuracy in weakly textured regions and additionally improves speed by rejecting unlikely plane hypotheses *a priori*.
- A graph-cut labeling method is presented for selecting the principal surface normal for each pixel of the image.

Variable Baseline/Resolution Stereo. (Chapter 5) In addition to recovering accurate depth estimates, it is also important to know how accurate those measurements are.

For stereo, depth uncertainty grows quadratically with depth, but it can be reduced by using higher image resolution or greater baseline (distance between cameras). In more detail,

- An analysis describes the advantages and disadvantages of increasing image resolution and/or baseline (camera distance) for improved depth precision.
- A view selection algorithm for balancing resolution and baseline guarantees constant depth precision throughout the reconstruction, independent of scene depth.
- The time complexity to achieve a target depth precision up to depth z is $O(z^3)$ with the proposed method, which improves over the basic view selection strategy that has $O(z^6)$ complexity.

Piecewise Planar and Non-Planar Stereo. (Chapter 6) To further improve the accuracy of the 3D reconstruction beyond the depth resolution limits of the stereo setup, higher-level knowledge must be used. Imposing a piecewise-planar model of an urban scene improves the reconstruction both quantitatively and qualitatively. However, not all elements of an urban scene are planar. In this chapter,

- Images and depthmaps are used to segment a scene into several planar regions as well as identifying non-planar regions. Non-planar regions are modeled by a general surface as given by the input depthmap.
- The segmentation is made possible in part by using a classifier which learns the appearance of planar surfaces from training data.
- Before segmentation, plane hypotheses are clustered across multiple views which allows a consistent piecewise-planar reconstruction to scale over videos of arbitrary length.

A Heightmap Model for 3D Reconstruction. (Chapter 7) Converting depth measurements into a surface is a non-trivial problem. Multiple redundant measurements must be fused, and surface topology (depth discontinuities) must be discovered. A volumetric approach is a natural way to handle these problems, but such methods are memory-intensive and thus have low resolution. A heightmap model is a more efficient approximation to the full 3D volume. In particular, I demonstrate the following:

- A heightmap model effectively models urban scenes and enforces strictly vertical walls and continuous surfaces without holes.
- The method, based on probabilistic occupancy, fuses multiple measurements while using little memory, running efficiently on the GPU, and producing low-complexity output models.
- Multiple layers can be used to model more general structure with a heightmap. Dynamic programing is used to compute the optimal positions of the layers for a single heightmap pixel.
- The heightmap method is also flexible enough to reconstruct scenes from photo collections gathered from the web.

1.3 Organization

The remainder of the dissertation is organized as follows. Chapter 2 describes how 3-dimensional structure can be recovered from images. The main issues are establishing correspondence across multiple views, understanding how that correspondence translates into a 3D measurement, and recovering the scene's surface from those measurements. Chapter 3 contains a literature review of urban 3D reconstruction and relevant stereo techniques.

Chapter 4 addresses the correspondence problem. The scene's urban structure is analyzed and multiple plane-sweeps are performed to correctly compute matching scores in the presence of slanted surfaces. The analysis of the scene's structure also improves the speed of the stereo method by focusing effort where the surfaces are likely to be found. This stereo matching method runs in real-time on the GPU.

Chapter 5 analyzes how the camera baseline and image resolution influence the precision of the depth measurements. A view selection algorithm takes advantage of the flexible baseline provided by video and guarantees constant depth resolution by allowing the baseline and image resolution to vary during the stereo computation. This method has superior time complexity compared to a fixed camera setup.

Chapter 6 improves the precision of the 3D reconstruction by selectively imposing a piecewise-planar surface model on the scene. This reconstruction goes beyond just measuring depth, and looks at the qualitative properties of the scene, segmenting it into planar and non-planar surfaces based on the learned appearance of such surfaces. A multi-view plane clustering step ensures the reconstruction scales to scenes of arbitrary size.

Chapter 7 presents a method for fusing all depth measurements into a single volume and extracting a compact multi-layer heightmap representation of the scene. Surfaces are extracted from the heightmap in such a way that vertical walls are strictly enforced. The method efficiently runs on the GPU, and the produced heightmaps are a compact way to store a 3D reconstruction of an urban environment.

Finally, Chapter 8 concludes with a summary and discussion of ideas for future work.

CHAPTER 2

3D Reconstruction

In this chapter, I will describe how 3D structure can be reconstructed from 2D images of a scene, and I will summarize the key challenges of 3D reconstruction addressed in this dissertation. These are multi-view **correspondence**, 3D measurement **geometry**, and **surface** estimation.

2.1 Camera Geometry

Before describing how 3D structure can be reconstructed from images, it is important to understand how images are formed. Most images studied in computer vision are formed by a *camera*. The name originates from *camera obscura*, literally *dark chamber*. It was so named because the device blocks out all light, except the light passing through a *pinhole*, which is then received by the *retina* that forms the image. Each point in the image is illuminated by the light energy incoming along the *viewing rays* originating at the image point and passing through the pinhole. Digital cameras use an array of sensors (CCD or CMOS) that convert this light energy at each point into a number called the *intensity*. The array of all intensities is called the *image*, and may contain multiple color channels (different wavelengths of light). Modern cameras also have lens systems which gather and focus the light; however, the mathematics of the pinhole camera are much simpler and are sufficient for this dissertation (once lens distortion is accounted for).

Figure 2.1 illustrates how images are formed under the pinhole camera model. Figure 2.2 shows geometrically how a 3D point $\mathbf{X} = [X \;\; Y \;\; Z]$ is projected onto a point $\mathbf{x} = [x' \;\; y' \;\; z']$ on the image plane. This projection can be expressed by the following formulas which can

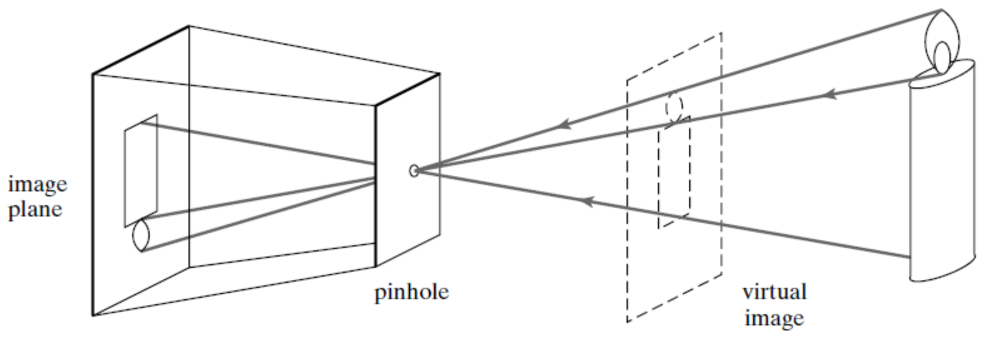


Figure 2.1: A pinhole camera. Image courtesy of Forsythe and Ponce (2002).

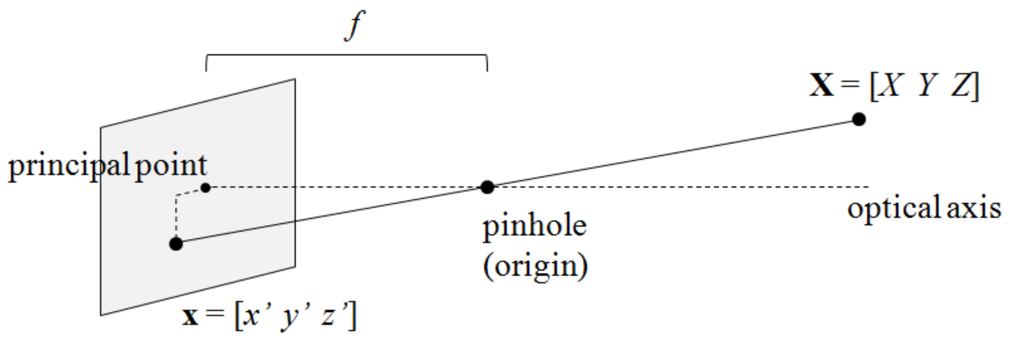


Figure 2.2: The pinhole camera model projects 3D point $\mathbf{X}$ onto point $\mathbf{x}$ in the image plane.

be derived from similar triangles:

$$x' = f\frac{X}{Z} \qquad y' = f\frac{Y}{Z} \qquad z' = f \tag{2.1}$$

where f is the focal length (distance of the image plane to the pinhole). Because the image plane and the scene are on opposite sides of the pinhole, f and Z will have opposite signs (typically f is negative and Z is positive), resulting in the inverted (flipped and mirrored) image shown in the Figure 2.1. It is simpler to work with the virtual image which is not inverted and where f is positive.

Note that all points $\lambda\mathbf{X}$ ($\lambda \neq 0$) will also project to $\mathbf{x}$ since the scaling factor cancels in Equation 2.1. This is equivalent to the fact that all points on the viewing ray project to the same point on the image plane, meaning that the distance or depth of $\mathbf{X}$ is lost under projection. 3D reconstruction is the process of recovering the depth of the scene and is the focus of this dissertation.

The fact that all points on the viewing ray project to the same point in the image plane can be expressed conveniently using homogeneous coordinates. In homogeneous coordinates, two points are considered equal if the vectors representing them are equal up to a non-zero scale factor. This requires vectors to have one more coordinate than the dimension of the space so that all points can be represented. As an example, a point $[3 \;\; 2]$ in Euclidean 2-space could be represented by the vectors $[3 \;\; 2 \;\; 1]$ and equivalently $[6 \;\; 4 \;\; 2]$ in homogeneous coordinates. In homogeneous coordinates, the image point is $\mathbf{x} = [x' \;\; y' \;\; z' \;\; w']$ (where $w' = 1$). We can scale $\mathbf{x}$ by Z to obtain

$$x' = fX \qquad y' = fY \qquad z' = fZ \qquad w' = Z. \tag{2.2}$$

Since the point is already assumed to be on the image plane, the coordinate z' is unnecessary, and therefore $\mathbf{x} = [x' \;\; y' \;\; w']^\mathsf{T}$. Using homogeneous coordinates on the image plane, projection can be expressed linearly:

$$\mathbf{x} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{X}. \tag{2.3}$$

Up until now, we have ignored some important details. For example, points on the the image plane are usually expressed in pixel coordintes where the origin is at the top left corner of the image and pixels are spaced one unit apart. Also, when multiple cameras are used (or the same camera at different positions/orientations), it is useful for each camera to have it's own coordinate system. The parameters that control how points in world coordinates map to pixel coordinates are called the camera *calibration*. The camera calibration can divided into intrinsic parameters and extrinsic parameters. Intrinsic parameters describe the internal properties of the camera, namely how points on the camera's image plane are mapped to pixel coordinates in the image. The extrinsic parameters describe the external properties of the camera, which are the camera's position and orienation in the world.

The intrinsic calibration can be expressed as a linear equation similar to Equation 2.3 but with addtional parameters:

$$\mathbf{x} = \mathbf{K}\mathbf{X} \quad \text{where} \quad \mathbf{K} = \begin{bmatrix} f & s & p_x \\ 0 & \alpha f & p_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.4}$$

The focal length f indicates the distance of the image plane to the pinhole measured in pixel units. Larger focal length results in more pixels per unit viewing angle and can be achieved either through magnification or with a higher resolution image. The other intrinsic parameters are skew s, typically equal to 0, and the aspect ratio α, typically equal to 1 meaning square pixels. The principal point (p_x, p_y) is the point in the image closest to the pinhole where the viewing ray (called the optical axis) is orthogonal to the image plane. For most cameras, this point is close to the center of the image.

Equation 2.4 is derived according to the pinhole model. However, modern cameras have lenses for gathering and focusing the light. Lenses introduce geometric distortion, which causes the viewing rays to deviate from the pinhole model. This effect is often significant and must be accounted for to obtain a good reconstruction. Most lens distortion can be modeled with the radial distortion model. Let (x', y') be the ideal projection of a point under the pinhole model. These are inhomogeneous coordinates obtained by dividing by the third component. Let (x, y) be the distorted point which is obtained as follows:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix} \left(1 + \kappa_1 r^2 + \kappa_2 r^4 + \ldots\right) \tag{2.5}$$

$$\text{where} \;\; r = \sqrt{(x - p_x)^2 + (y - p_y)^2}. \tag{2.6}$$

The parameters $\kappa_1, \kappa_2, \ldots$ are part of the intrinsic parameters of the camera. Once they are calibrated, radial distortion can be removed from the images as a preprocess.

The extrinsic parameters define the position and orientation in the world. They are $\mathbf{R}$ and $\mathbf{t}$ where $\mathbf{R}$ is a rotation matrix that defines the orientation of the camera, and $\mathbf{t}$ is the

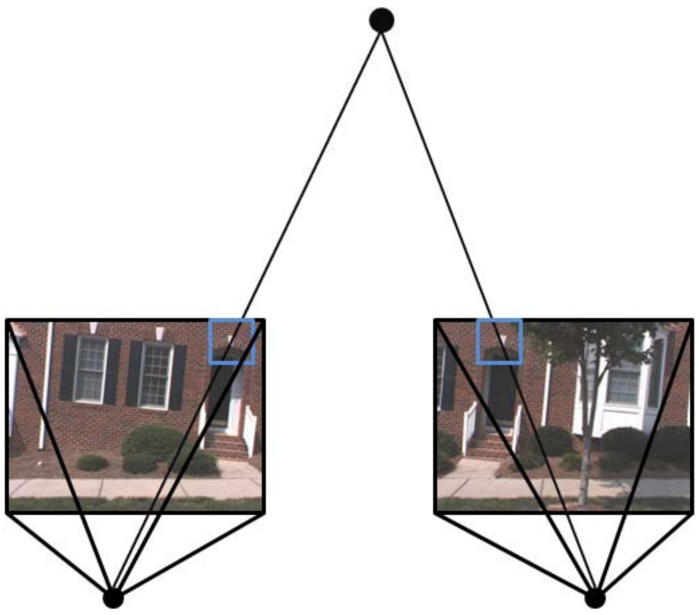


Figure 2.3: A 3D point reconstructed from correspondence.

translation vector that defines the position of the camera. The location of the pinhole, also called the camera center, is $\mathbf{c} = -\mathbf{R}^{\mathsf{T}}\mathbf{t}$. Using homogeneous coordinates for the 3D point $\mathbf{X} = [X \;\; Y \;\; Z \;\; 1]^{\mathsf{T}}$, and once radial distortion has been accounted for, we can express the camera as a 3×4 matrix:

$$\mathbf{x} = \mathbf{P}\mathbf{X} \quad \text{where} \quad \mathbf{P} = \mathbf{K}\,[\mathbf{R} \;\; \mathbf{t}]. \tag{2.7}$$

2.2 Reconstruction from Correspondences

3D reconstruction depends on determining correspondence between images. Figure 2.3 shows a 3D point that has been reconstructed from a point correspondence in two images. This involves two steps: obtaining the corresponding points in the image planes, and determining their 3D positions. If the camera parameters are known, the 3D point can be obtained by intersecting the viewing rays of the corresponding points as shown in the figure. We will first discuss correspondence and reconstruction in general, and then we will consider the case where the camera parameters are known.

When searching for correspondences, it is useful to have a metric to determine the likelihood of two or more points corresponding. We will call this metric the matching score. Matching scores are based on the assumption that corresponding points have a similar appearance. When a 3D point is imaged in multiple views, it is assumed that the images of that point will have the same intensity. This is true if the camera exposure remains constant, and if the surface of the point is Lambertian, meaning that it reflects light equally in all directions. Given two candidate correspondences, their dissimilarity can be determined by absolute or squared differences. Single pixels are too ambiguous to determine a reliable match by differencing, so the surrounding patch of pixels are often differenced as well, leading to the sum of absolute differences (SAD) and sum of squared differences (SSD) matching scores. The method of normalized cross-correlation (NCC) first normalizes the patches to achieve invariance to linear changes in intensity. Other metrics such as the rank and census transforms achieve even greater invariance by considering only relative intensity information (Hirschmuller and Scharstein, 2008).

Exhaustively searching the images for correspondence would be time consuming. This would also lead to many false correspondences since many points do not have a distinct appearance. Instead, candidate point correspondences can be determined by extracting salient *corners* from the images and exhaustively comparing the two sets. The Harris corner method (Harris and Stephens, 1988) is one example of this matching method. Tracking methods such as the Kanade-Lucas-Tomasi (KLT) algorithm (e.g. Shi and Tomasi, 1994), track the motion of salient feature points assuming the motion is small, as is often the case in video. If the camera parameters (intrinsic and relative extrinsics) are known, the search space can be reduced considerably, and exhaustive comparisons for all pixels can be performed to achieve dense correspondence. This dense correspondence problem is often referred to as stereo. Stereo correspondence will be the focus of this dissertation, but first we will briefly discuss the case where the camera parameters are not known.

Even if the camera parameters are not known, then by the projective reconstruction theorem (Hartley and Zisserman, 2000), both camera parameters and the 3D points can be solved up to a projective ambiguity (an arbitrary 4×4 matrix transformation). This is known as a projective reconstruction. If some parameters are known, or are known to be fixed across views, this ambiguity can be reduced. Even when all camera intrinsics are known, there is still a scale ambiguity. This is known as a metric reconstruction. Using correspondences to solve for the camera parameters and 3D points is a fundamental problem in computer vision known as structure from motion (SfM). The SfM system used in Urbanscape incorporates GPS and inertial sensors to eliminate the scale ambiguity and make the camera parameter estimation extremely robust. This is known as a Euclidean reconstruction. It is also *geo-located* because the coordinates of the reconstruction coincide with standard Earth coordinate systems such as Universal Transverse Mercator (UTM). The methods presented in this dissertation use the SfM results of Urbanscape and focus on the case where the camera parameters are known.

When the camera parameters are known and the scene is static, the correspondence search domain is reduced from 2 dimensions (the whole image) to 1 dimension (a single line in the image called the epipolar line). Consider a 3D point $\mathbf{X}$ and its image $\mathbf{x}$ shown in Figure 2.5a. To reconstruct $\mathbf{X}$ from $\mathbf{x}$, we only need to consider all points on the viewing ray. If the camera parameters are known, then the viewing ray is also known, and is given by the following formula:

$$\mathbf{X}(\rho) = \mathbf{c} + \rho \mathbf{R}^T \mathbf{K}^{-1} \mathbf{x} \quad \text{where } \rho > 0 \tag{2.8}$$

Projecting the ray $\mathbf{X}(\rho)$ into another image yields the epipolar line for point $\mathbf{x}$. This is only a half-line because impossible points behind the camera ($\rho \leq 0$) are discarded. In general, two images can be *rectified* so that the epipolar lines are corresponding rows in the image. Two rectified images are shown in Figure 2.4. In this case, correspondences have the same y coordinate, and the difference between x coordinates is called the *disparity*.

More than two views cannot be rectified in general, but a ray search can again be considered. Figure 2.5b shows points along the viewing ray projected into multiple views. This leads to a simple multi-view stereo algorithm found in Algorithm 2.2. For every point on a ray, the point is projected into all views, and matching scores are computed and summed. The best matching point on each ray is then chosen resulting in a depthmap.

Algorithm 1 A simple method for computing a depthmap.

```
for all points $\mathbf{x}$ in the image do
  $D_\mathbf{x} \Leftarrow 0$
  $E_\mathbf{x} \Leftarrow \infty$
  for all $\rho = \rho_{min}$ to $\rho_{max}$ do
    $\mathbf{X} \Leftarrow \mathbf{c} + \rho\mathbf{R}^T\mathbf{K}^{-1}\mathbf{x}$
    $E \Leftarrow 0$
    for all views $i$ do
      // project $\mathbf{X}$ into view $i$
      $\mathbf{x}_i \leftarrow \mathbf{P}_i\mathbf{X}$
      compute matching score $E_i$ between $\mathbf{x}$ and $\mathbf{x}_i$
      $E \Leftarrow E + E_i$
    end for
    if $E < E_\mathbf{x}$ then
      $D_\mathbf{x} \Leftarrow \rho$
    end if
  end for
end for
```

2.3 Stereo Correspondence and Reconstruction

There are several issues with computing matching scores that are important to stereo. Many matching scores commonly used in stereo are computed from a neighborhood of pixels surrounding the candidate point. The patch of pixels surrounding the point is called the matching window (or correlation window). Rectangular neighborhoods are commonly used for simplicity and ease of implementation. It is assumed that the center pixel and its neighbors will have the same appearance from view to view. If the window overlaps a depth discontinuity or if the surface is highly slanted, then this will not be the case. Figure 2.6 shows several examples of corresponding points and their matching windows. Matching scores of points near depth discontinuities lead to a phenomenon called boundary overextension (Okutomi and Kanade, 1993) or foreground fattening. This problem is prevalent in real-time stereo methods which use window-based matching due to its speed and parallelizability, and some real-time methods for handling overextension have been proposed (Gallup et al., 2009; Hirschmüller et al., 2002).

The mismatches due to slanted surfaces can be corrected if the the slant of the surface is known (or solved for). Figure 2.7 shows two views of a slanted surface. The mismatch due to the slant has been corrected by warping the image with a *homography*. A homography

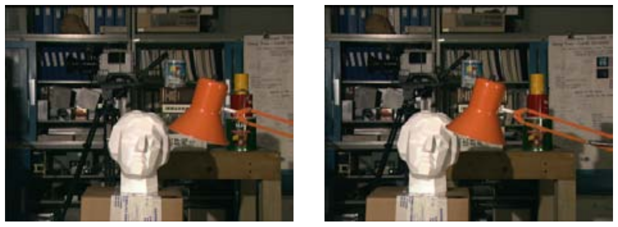


Figure 2.4: Two rectified images.

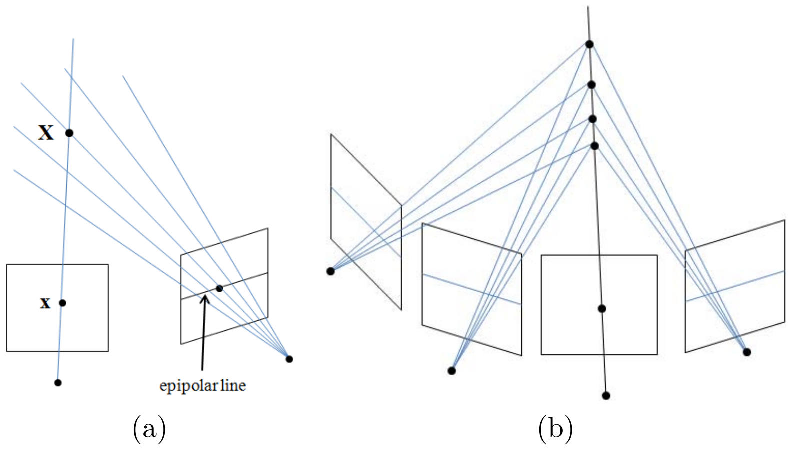


Figure 2.5: Correspondence search geometry. (a) Two view correspondence search. (b) Multi-view correspondence search.

is a projective mapping between two spaces of the same dimension, and can be used to map points between planes for example (Hartley and Zisserman, 2000). Because the images are planes and the surface is planar, a homography can be used to map one image to the other, correcting the perspective distortion induced by the slant. If camera and surface plane parameters are known, then using homogeneous coordinates the homography can be represented as a 3×3 matrix (linear transformation). Assuming the first camera has no rotation (identity) and is located at the origin, the homography is given by the following formula:

$$\mathbf{H} = \mathbf{K}_1 \left(\mathbf{R}_1 - \frac{\mathbf{t}_1 \pi_{\mathbf{n}}{}^T}{\pi_d} \right) \mathbf{K}_0^{-1}. \tag{2.9}$$

where $\mathbf{K}_0$,$\mathbf{K}_1$ are the intrinsics for the first and second camera, $\mathbf{R}_1$ is the rotation of the second camera, $\mathbf{c}_1$ is the center of the second camera, $\pi_{\mathbf{n}}$ is the plane normal, and π_d is the distance of the plane to the origin. Chapters 4 and 6 use homographies to compute

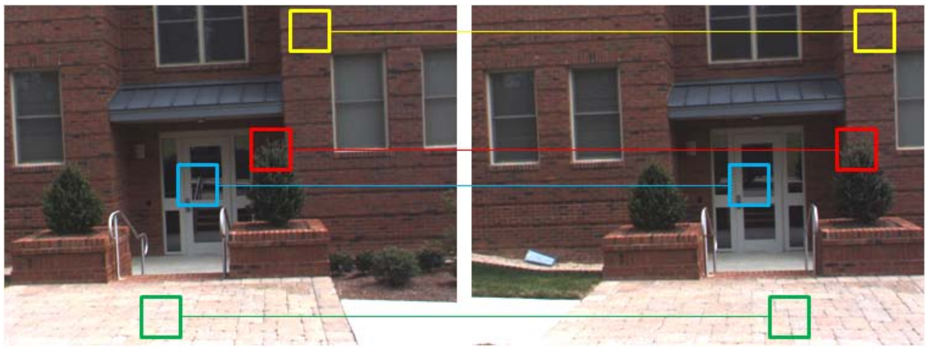


Figure 2.6: Difficulties with matching. (Red) The matching window overlaps a depth discontinuity. (Green) The matching window is affected by perspective distortion. (Blue) The matching window contains a reflection. (Yellow) A good match is obtained.

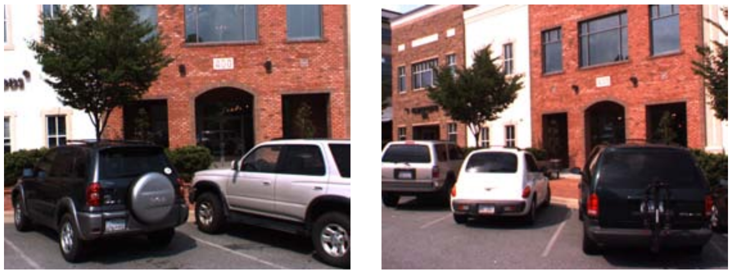
matching scores for plane hypotheses in the scene.

One way to avoid the overextension and slant problem is to compute matching scores from single pixels. Without the supporting neighborhood, such matches become extremely ambiguous and the solution must be regularized. A surface smoothness prior can be imposed in the form of an energy functional:

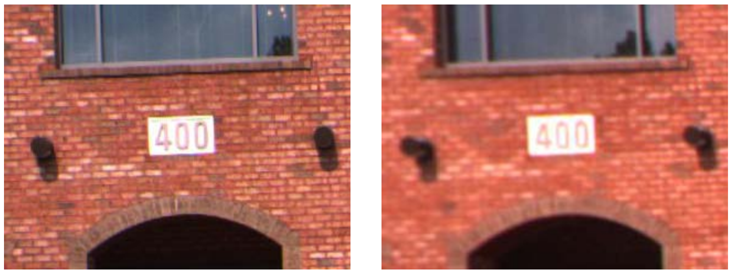
$$E(D) = \sum_{p \in \mathcal{I}} E^{p}_{data}(D(p)) + \lambda \sum_{(p,q) \in \mathcal{N}} E^{(p,q)}_{smooth}(D(p), D(q)). \tag{2.10}$$

$\mathcal{I}$ is the set of all pixels in the image, and $\mathcal{N}$ is the set of all neighboring pixels. The goal is to find a depth or disparity map D which minimizes $E(D)$. E_{data} encodes the matching scores and E_{smooth} penalizes large differences in neighboring values. λ controls the relative weight of the two terms. Algorithm 2.2 in fact solves a special case of Equation 2.10 where $\lambda = 0$, thus ignoring the smoothness term. Stereo methods which minimize only the data term are typically called *local* while methods that use a pairwise smoothness term are typically called *global* because the solution at each pixel depends on all other pixels through the pairwise interactions. Global methods yield better results, but solving the global energy is difficult, whereas the local energy can be solved exactly and efficiently especially due to the independence among pixels (Scharstein and Szeliski, 2002).

The sampling of points along the ray in Algorithm 2.2 is also an important consideration. Points should be chosen so that their projections lie less than one pixel apart. This is to



Two views of a surface distorted by perspective.



Close-up after being corrected by a homography.

Figure 2.7: Surface slant can be corrected with a homography if the surface plane is known.

prevent aliasing in the matching cost function, essentially resulting in missing the correct matching location. The image resolution and camera geometry determine the number of samples required (Szeliski and Scharstein, 2004), which has important time complexity ramifications as discussed in Chapter 5.

Occlusion is also a challenge. The correct 3D point **X** may not produce a good matching score if it is projected into a view in which the point is occluded, i.e. another surface makes **X** invisible in that view. Ignoring these occlusions results in unreliable matching scores and errors in the depthmap. Methods for handling occlusions usually entail either explicitly identifying occluded views or combining matching scores in a way that is robust to these outliers (Kang et al., 2001).

2.4 Reconstruction Accuracy

A critical part of 3D reconstruction is to understand the accuracy of the reconstruction. More formally, we will discuss the reconstruction in the following terms:

- **measurement:** The reconstruction obtained using an algorithm. For example, stereo computes a depth measurement for each pixel.

- **ground truth:** The true value of the reconstruction. For example, the true depth value for each pixel.
- **error:** The difference between the measurement and ground truth (signed or absolute).
- **uncertainty:** The expected distribution of error, derived from assumptions or from calibration.
- **accuracy:** Used informally to describe the error of the reconstruction. Greater accuracy means less error.

Reconstruction error is due to many sources: error in the camera parameters both intrinsic and extrinsic, and especially error in the correspondence estimation. Correspondence error is proportional to the image sampling rate (pixels) and higher resolution results in more accurate depth estimates. To illustrate how correspondence error translates into 3D position or depth error, consider two rectified views, shown in Figure 2.8. In the figure, z is the depth, b is the baseline or distance between the cameras, f is the focal length measured in pixels, and d is the disparity also measured in pixels. By similar triangles we have $z = -bf/d$. The disparity estimate of a correspondence has some error which we will denote ϵ_d. This translates into depth error as follows:

$$\begin{aligned} \epsilon_z &= \frac{bf}{d} - \frac{bf}{d + \epsilon_d} \\ &= \frac{z^2 \epsilon_d}{bf + z\epsilon_d} \\ &\approx \frac{z^2}{bf} \cdot \epsilon_d. \end{aligned} \tag{2.11}$$

The final step is obtained by taking the first order Taylor series approximation about $\epsilon_d = 0$. The depth error can be broken up into *correspondence error*, ϵ_d, and the *geometric factor*, $z^2/(bf)$. The geometric factor depends on the geometry of the stereo setup: the baseline, focal length, and distance to the point. Here we see that depth error is a quadratic function

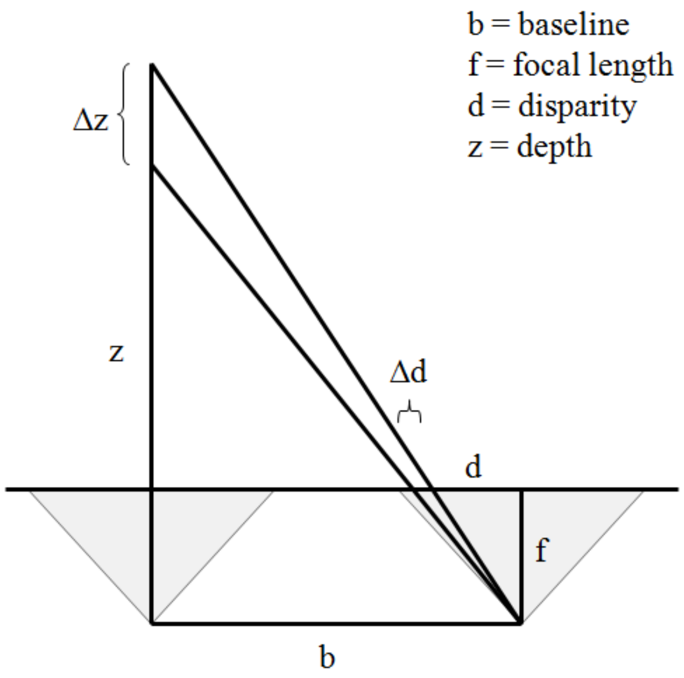


Figure 2.8: Depth error as a function of disparity error.

of the depth z of the point. This means there is a substantial difference in accuracy between objects in the near and far ranges of the reconstruction. But, as is shown in Chapter 5, the baseline and focal length can be adjusted to control the geometric factor to improve the accuracy.

2.5 Surface Reconstruction

The next step is to convert depth measurements into a surface. Whereas depth measurements represent isolated points in 3D, a surface is a 2 dimensional submanifold of the 3D space, with the important distinction being that a surface has a notion of neighborhood or connectedness between points. Inferring surface topology from a point cloud is a fundamental problem. It is addressed in general terms by Kazhdan et al. (2006), but here we will address it specifically for 3D reconstruction from images. Image-based approaches use the implied topology (adjacent pixels) of the depthmap, but depth discontinuities must be determined. This can be done somewhat effectively by simple thresholding. More advanced approaches exist such as Birchfield and Tomasi (1999a). Another strategy for surface reconstruction is volumetric. In such approaches, depth measurements are used to compute the occupancy of every point in the volume, and then the surface is extracted as the boundary between and occupied and unoccupied space (Zach et al., 2007). For an overview of other surface reconstruction approaches, see Fabio (2003).

2.6 Summary

Thus 3D reconstruction can be performed in three steps:

- **Correspondence**: Correspondences are established across multiple views.
- **Geometry**: 3D points and possibly camera parameters are computed from the correspondences.
- **Surface**: A surface is reconstructed from 3D point measurements.

In dense reconstruction (the focus of this dissertation), camera parameters are considered known, but they still affect point positions and especially their uncertainty. Note that the term "steps" applies only loosely. In fact, different algorithms may perform these steps in different order, and steps can be combined or omitted. For example, while structure from motion performs these steps in the order described, the stereo algorithm presented in Algorithm 2.2 reverses the correspondence and geometry steps. Points along the ray are considered and projected into the images according to the camera geometry. Then correspondences are determined by examining the matching scores. Finally, some methods adopt a surface-centric approach, hypothesizing a surface, projecting it into the images according to the geometry, and then evaluating the induced correspondences in order to update the surface (Vu et al., 2009). Regardless, each step plays an important role in 3D reconstruction.

Each of the remaining chapters presents a method which targets one or more of these steps.

- Chapter 4 presents a multi-sweep stereo approach which aims to improve **correspondences** by correctly handling slanted surfaces prevalent in urban scenes. This addresses the *correspondence error* in Equation 2.11.

- Chapter 5 analyzes the camera **geometry** related to 3D reconstruction from video and proposes a method to bound depth uncertainty and improve time complexity. This addresses the *geometric factor* in Equation 2.11.

- Chapter 6 reduces **correspondence** error by inferring **surface** properties such as planarity.

- Chapter 7 uses a heightmap model to fuse multiple depth measurements and recover a **surface**.

CHAPTER 3

Related Work

Now that I have given an overview and summary of 3D reconstruction, I will review the body of literature relevant to this dissertation. First I will discuss papers related to the 3D urban modeling problem in general. Then I will review the stereo literature in general. Finally I will narrow the focus to approaches that use stereo for street-level imagery.

3.1 Urban Reconstruction

Urban 3D reconstruction has been performed in computer vision and photogrammetry using various techniques and modalities. Range data sources include satellite, aerial, and terrestrial, and modalities include photography, light detection and ranging (LiDAR), and other active ranging techniques such as synthetic aperture radar (SAR). Processing can either be automatic or user-assisted (semi-automatic).

City reconstruction from aerial imagery has long been one of the problems studied in photogrammetry. An overview of automatic and semi-automatic methods was given by Förstner (1999). Semi-automatic methods such as that of Gulch et al. (1999), seek to combine the strengths of humans and computers. The user drives the reconstruction by selecting and grouping the important point or line features, and the computer refines their locations based on the images. A notable automatic method is that of Baillard et al. (1999). Line segments are extracted and matched over multiple aerial views using the trifocal tensor (Hartley and Zisserman, 2000), and piecewise-planar rooftops are reconstructed. After the lines are triangulated in 3D, the orientation of the roof planes is computed by maximizing the matching score across views. Roof segments are then delineated by grouping and intersecting the planes.

The work of Zebedin et al. (2006) is a more recent example of a fully automatic method for aerial imagery. Whereas semi-automatic methods prefer to use only a few images to reduce the amount of user interaction required, fully automatic methods can take advantage of a large number of views. The high degree of overlap and redundancy improves the robustness of automatic methods and allows this method to perform not just line matching but fully dense stereo matching. Classification performed on multispectral images allows buildings to be extracted and reconstructed in 3D. The follow-up work of Zebedin et al. (2008) focuses on converting the buildings into simplified models composed of planes and surfaces of revolution. This type of classification and planar reconstruction is one of the inspirations for the method presented in Chapter 6. Since the target of our method is street-level video, the problem is somewhat different, and our work deals with issues such as scaling the reconstruction consistently over the video.

Light detection and ranging (LiDAR) is an alternative to purely image-based methods. LiDAR measures distance by emitting laser pulses and measuring the time between transmission and detection of the return signal. LiDAR is typically more robust and accurate than stereo. Several methods have been developed to process aerial-based LiDAR data for city reconstruction. Morgan and Tempfli (2000) and You et al. (2003) are automatic and semi-automatic examples of methods that reconstruct buildings from LiDAR range data. Buildings are segmented and the range measurements are refined. The final surface models are obtained by fitting geometric primitives to the data.

Compared to terrestrial data, the aerial perspective provides greater coverage and entire cities can be captured with relatively few images. However, the data contains mostly rooftops, and building facades are occluded, heavily distorted, and/or captured with low-resolution. Ideally, the large-area aerial perspective should complement a high-detail terrestrial perspective.

LiDAR technology has also been used to reconstruct cities from ground level. Früh and Zakhor (2004) developed an automobile-mounted LiDAR system for capturing detailed city models from the streets. A horizontally aligned LiDAR scanner measures a horizontal slice of the scene which is used to compute the vehicle motion by iterative closest point (ICP) registration (Besl and McKay, 1992). Path estimation errors are corrected by aligning the reconstruction with aerial views. A vertically aligned LiDAR scanner captures the building facades as the vehicle drives through the city streets, and those scans can be fused with and aerial-based LiDAR reconstructions to produce complete models (Früh and Zakhor, 2003).

Despite the robustness and accuracy of active ranging methods like LiDAR, performing reconstruction from only images is still a major goal. Photographs and video are acquired passively using the light naturally present in the scene. LiDAR in comparison requires more power to operate, and the energy emitted poses a safety concern (even though LiDAR can be rated eye-safe). Furthermore, cameras are relatively inexpensive and ubiquitous. For city-scale urban reconstruction, cost is an important concern due to the massive scale of the problem.

Several methods have been proposed for modeling architecture from photographs or video. The *Facade* system presented by Debevec et al. (1996) is a user-assisted or semi-automatic approach. Users indicate key edges in several images which are then formed into blocks. Edge positions and the 3D structure are refined automatically. The user can also specify relative position and symmetry constraints to further aid the reconstruction. Finally, model-based stereo is performed to estimate small deviations from the block model. The work of Sinha et al. (2009) presents a similar system which uses vanishing points to further assist the user in specifying the model. By automatically aligning polygon edges to vanishing directions, the user can create the model in fewer steps and with greater precision. Another semi-automatic system is presented by Xiao et al. (2008). First the urban scene is fully automatically reconstructed as a collection of rectilinear blocks. Then several simple tools are provided to allow the user to correct the mistakes made by the system.

While having a user in the loop can increase the robustness of the system, semi-automatic methods become impractical for large-scale reconstruction problems. Some of the earliest ground-based automatic urban reconstruction works include the MIT City Scanning Project (Teller, 1998). A mobile robotic platform is used to capture thousands of photographs from the grounds of the MIT campus, and the camera parameters are automatically estimated using structure from motion and GPS data. Vertical facades are extracted and texture-mapped (Coorg and Teller, 1999) and facade details such as windows are recovered (Wang et al., 2002).

Schindler et al. also recovered camera poses and sparse edge structures in the 4D Cities project (Schindler and Dellaert, 2004; Schindler et al., 2006). Methods for edge matching and line-based structure from motion are presented. Also, given a database of historical photographs of Atlanta, the temporal sequence of the photos and temporal changes in the city were reconstructed (Schindler and Dellaert, 2010).

Examples that use explicit architectural models include the work of Dick et al. (2001). The method uses a bayesian model describing architectural style, window and door dimensions, and parameters of other architectural features. Priors are constructed from rules found in architectural texts, and the model is solved using a Monte Carlo method. Müller et al. (2007) developed a method for procedural modeling from photographs of architectural scenes. Similarly, Venagas et al. (2010) assume a Manhattan-world (3D rectilinear) structure and develop a grammar that describes constraints on building shapes. A model is generated from the grammar that fits the images according to edges and multi-view photoconsistency.

This related work focuses on modeling only the architecture of the scene. In contrast, the methods presented in this dissertation are designed to model architecture as well as the myriad objects found in urban scenes. The stereo and fusion techniques in this thesis are developed to take advantage of urban scene properties like planarity, orthogonality, and verticality, but these techniques are general enough to reconstruct objects of any shape.

3.2 Stereo

Recovering the shape of the scene from images is generally referred to as *stereo*. Stereo reconstructs shape by establishing dense correspondences between views, a problem which is ill-posed since an infinite number of shapes can generate the same images. Stereo can use two or more views. The two view case is called *binocular* and the case of three or more views is called *multi-view*. Scharstein and Szeliski (2002) give a survey and evaluation of binocular stereo methods. Global methods (see Equation 2.10) are among the top performers and include graph cuts (Boykov et al., 2001), belief propogation (Felzenszwalb and Huttenlocher, 2004a), and continuous methods (Pock et al., 2008). These methods solve the energy minimization problem only approximately or require the energy to take a special form. Furthermore, computation takes seconds to minutes, and even the fastest variants (Yang et al., 2006) take seconds or can only achieve real-time performance at low resolution. Methods such as dynamic programming (Belhumeur, 1996) and semi-global matching (Hirschmuller, 2008) simplify the global energy by considering pairwise smoothness constraints only along 1-dimensional paths. These represent a good balance between quality and efficiency; however, these methods are still much slower than local methods.

This dissertation employs a local method in Chapter 4, semi-global matching in Chapter 5, and a graph cut method in Chapter 6 and optionally in Chapter 4. Since the stereo method in Chapter 4 must be applied to every video frame, a fast local method is used. In Chapter 5 semi-global matching is used to avoid the need for matching windows and places the emphasis on the *geometric factor* of stereo precision. The formulation in Chapter 6 results in an energy function which is more difficult to optimize, so a graph-cut method is used. Finding a faster solution is a matter of future work, but note that this graph-cut method is used only on a fraction of video frames, and all frames can be processed in parallel.

In the same spirit of the binocular stereo evaluation, Seitz et al. (2006) compare and evaluate multi-view stereo methods. Space carving (Kutulakos and Seitz, 2000) is among the earliest multi-view stereo methods. A shape which matches the input images is called *photo-consistent* and the largest photo-consistent shape is called the *photo-hull.* Starting with an initial shape, the method carves away points that are not photo-consistent, guaranteeing at each step that the shape contains the photo-hull. At convergence, the remaining shape is the photo-hull. Determining whether a point is photo-consistent or not is of course a difficult problem due to image noise and calibration errors. Thus modern methods rely on soft photo-consistency measures, and the term photo-consistency has become synonymous with matching score.

Among the top multi-view stereo algorithms are Vu et al. (2009) and Furukawa and

Ponce (2008). These methods use the most accurate matches to initialize the reconstruction which is then refined either by region growing or mesh deformation. Another powerful approach, which is also GPU-friendly, is that of Zach et al. (2007). Depthmaps are computed for each view using a local stereo method (plane-sweep). The depth measurements are then used to estimate the occupancy of each point in a volume using total variation as a regularizer. The method in Chapter 7 also computes volumetric occupancy, but it introduces a 2.5D layer constraint that allows for a simpler, more memory efficient, and faster computation.

A popular technique for multi-view stereo is plane-sweeping. Plane sweeping simplifies the multi-view correspondence search because it avoids the need for rectification. Plane-sweeping is similar to the ray search described in Algorithm 2.2, except that the rays are searched plane by plane. Thus it can be seen as a plane that is swept through space, testing each plane location according to photo-consistency. This is described in more detail in Chapter 4. Testing points as plane hypotheses can be done efficiently and in parallel on the GPU (Yang and Pollefeys, 2005). The advantages of multi-view matching are described by Okutomi and Kanade (1993). Multiple views not only increase robustness to noise, but reduce the chance of mismatches due to repetitive structures. One question in plane sweeping is what planes to test, or what direction to sweep. Zabulis and Daniilidis (2004) tested all possible orientations at each point on the surface which required a cluster of computers to compute. The method presented in Chapter 4 avoids the need to test all orientations by computing the dominant surface normals and sweeping planes in those directions.

The method in Chapter 4 takes advantage of urban structure, namely the existence of a few dominant surface normals, to reduce the number of planes to be tested. Scene structure, particularly urban structure, was also used by Werner and Zisserman (2002). Line and point features are searched to detect planes and dominant surface normals, and plane locations are refined using plane-sweeping. Furukawa et al. (2009a) used an even more strict Manhattan-world assumption where the scene is represented by purely orthogonal planes. Less restrictive is the method of Sinha et al. (2009) which reconstructs the scene as a piecewise planar surface. The method targets urban scenes, and the strong planarity assumption helps with surfaces that have low texture and surfaces like glass that are difficult to match. The method of Bleyer et al. (2010) is even more general and segments the scene into planar and b-spline surfaces. Similarly, the work of Labatut et al. (2009) reconstructs a scene as a collection of plane, cylinder, cone, and sphere surfaces. The method presented in Chapter 6 also fits simple plane primitives to the scene, but allows parts of the scene to be modeled by a general depthmap surface. The method takes advantage of structured texture and color cues to segment man-made planar objects from non-planar objects like vegetation.

3.3 Street-Level Stereo Reconstruction

Works that are most related to this dissertation are those that reconstruct urban scenes from street-level images using stereo. The work of Cornelis et al. (2008) uses a U-shaped corridor model to reconstruct streets lined with vertical facades. Images are rectified to the vertical direction, meaning that image columns correspond to vertical lines. The strict vertical facade assumption allows all pixels in a column above the ground plane to be assigned the same depth, and the scene can be reconstructed with dynamic programming. The model cannot support non-facade objects like cars that are frequently parked in front of buildings, so the method integrates car and pedestrian detectors to remove these artifacts. The heightmap method in Chapter 7 is partially inspired by this vertical corridor approach, but the heightmap, and especially the multi-layer heightmap, also allows more general objects to be reconstructed.

The work of Micusik and Kosecka (2009) use images from Google Street View datasets and perform a piecewise-planar reconstruction of the scene. Images are segmented into superpixels, and each superpixel is assigned a depth and one of three urban scene normals. This is similar in spirit to the method presented in Chapter 4 which in fact serves as inspiration for their method.

The work of Xiao et al. (2008) demonstrates a system for urban street-side reconstruction which identifies facades and segments them into rectilinear patches. The position and orientation of each planar patch is obtained by minimizing multi-view photo-consistency. The method requires user interaction at various steps to produce high-quality results; however, a follow-up paper (Xiao and Quan, 2009) presents a similar fully automatic system. This system also uses a texture- and color-based classifier to segment buildings from ground, sky, and vegetation. Only the buildings are reconstructed using the rectilinear model. The piecewise planar method from Chapter 6 also uses a classifier in the same spirit. However, the class likelihood is combined with photoconsistency to simultaneously segment and reconstruct the scene, and non-building surfaces are reconstructed as well.

Irschara et al. (2007) show a handful of compelling results of scenes modeled with the volumetric stereo method of Zach et al. (2007). However, the method is not shown to scale beyond the size of the volume that can fit in memory. In contrast, scalability is a major concern in this dissertation.

CHAPTER 4

Plane-Sweeping Stereo with Multiple Sweeping Directions

4.1 Introduction

I will now introduce a plane-sweep stereo algorithm that is designed to take advantage of the structure in urban scenes and achieves real-time performance. A typical urban scene may contain a ground plane, two vertical, orthogonal facade planes, and other non-planar objects. Our method computes the three planar surface normals and a plane-sweep is performed in each direction. This approach is particularly accurate and efficient for typical urban scenes, and also preserves the ability to perform reconstructions of general 3D shape.

Due to the enormous amount of video data required to reconstruct entire cities, the stereo method needs to operate as efficiently as possible. Plane-sweep stereo is ideal for efficient implementation due to its simplicity and parallelizability (Collins, 1996; Yang and Pollefeys, 2003). The primary operation of the algorithm, rendering images onto planes, is an operation at which the GPU is particularly adept.

Like other real-time stereo algorithms, plane-sweeping requires a matching window. As discussed in Chapter 2, matching windows are less accurate for slanted surfaces, often resulting in staircase artifacts. The typical approach is to sweep a plane which is *fronto-parallel* (parallel to the reference view's image plane), and so surfaces which are not fronto-parallel will be matched poorly. However, plane-sweeping can correctly handle these slanted surfaces by sweeping a plane which has a matching surface normal. In our approach, we perform multiple plane-sweeps, where each plane-sweep is intended to reconstruct planar surfaces having a particular normal. Our algorithm consists of three steps. First, we identify

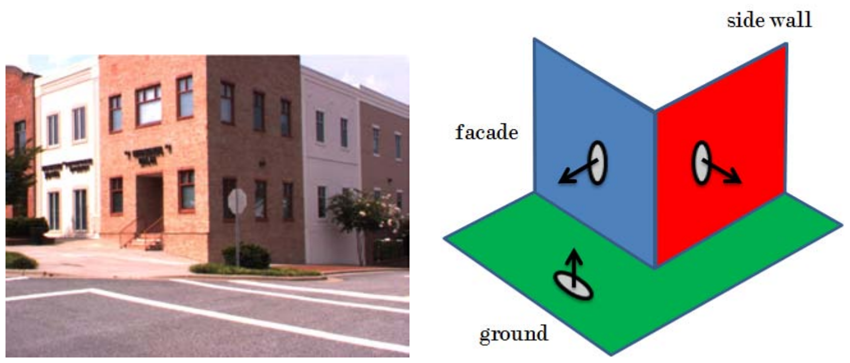


Figure 4.1: The major surface normals of an urban scene: ground, facade, and side wall.

the surface normals of the three most prevalent planar surfaces in the scene, i.e. the ground, facade, and side wall planes (see Figure 4.1). These normals are obtained by analyzing the 3D points obtained from structure from motion and by exploiting urban scene properties such as planarity, orthogonality, and verticality. Second, we perform a plane-sweep for each surface normal, resulting in multiple depth candidates for each pixel in the final depthmap. Third, we select the best depth/normal combination for each pixel using a simple best-cost approach or, optionally, a more advanced three-label graph cut which takes smoothness and integrability into account.

Additionally, we incorporate priors obtained from sparse point correspondences into our depth estimation. This aids in areas with little texture and produces a smoother result. We can also significantly reduce computation time by sweeping planes only in those regions with high prior probability according to the sparse data. Finally, we evaluate our algorithm on several scenes, and demonstrate the accuracy gained over the basic plane-sweeping algorithm.

The rest of the chapter proceeds as follows. Section 4.2 reviews some related work specific to plane-sweeping, slanted surfaces, and urban reconstruction. Section 4.3 describes plane-sweep stereo and points out the inherent problem of matching windows and plane orientation. Section 4.4 proposes our solution of using multiple sweeping directions, and section 4.5 presents experimental results.

The work in this chapter was presented in Gallup et al. (2007).

4.2 Related Work

The plane-sweeping algorithm was introduced by Collins (1996) as a way to perform matching across multiple images simultaneously without the need for rectification. The approach was originally targeted at the reconstruction of sparse features. Yang et al. (2002) implemented the plane-sweeping stereo algorithm on the GPU, achieving real-time dense depth estimation.

An approach for reconstructing buildings was described by Werner and Zisserman (2002) who use sparse point and line correspondences to discover the ground and facade planes. When there is insufficient information to locate a plane, they sweep a hypothesized plane through space to determine the position which best matches the images. We also use sparse features to obtain information about the scene's planar structure. However, rather than estimating the location of a sparse set of planes, we seek to compute a depth estimate for every pixel in the reference view. Our algorithm is therefore able to reconstruct objects such as trees and cars that do not fit the planar model.

Several stereo algorithms explicitly handle slanted surfaces. Burt et al. (1995) advocates pre-warping the images to a reference plane, such as the ground, before performing binocular stereo. In this way, they achieve greater accuracy as well faster computation due to the reduced disparity range. Our approach essentially achieves this pre-warping by adjusting our sweeping plane to be parallel to the expected planar surfaces in the scene. Birchfield and Tomasi (1999) cast the problem of stereo as image segmentation followed by the estimation of affine transformations between corresponding segments. Processing iterates between segmentation and affine parameter estimation for each segment using graph cuts (Boykov et al., 2001). The energy function does not favor constant disparity surfaces, but accounts for affine warping and thus slanted surfaces. Ogale and Aloimonos (2004) point out that if scene surfaces exhibit horizontal slant, then M pixels on an epipolar line necessarily correspond to N pixels in the other image. Therefore, requiring a one-to-one correspondence for every pixel results in labeling $|M-N|$ pixels as occluded. These pixels that are interleaved with matched pixels, however, are visible in both images, just not at integer coordinate positions. An algorithm based on dynamic programming is proposed to obtain correspondences between segments of scanlines rather than pixels.

Zabulis and Daniilidis (2004) explicitly addressed non-fronto-parallel surfaces by performing correlation on 3D planes instead of the image plane. Thus, correlation kernels can be aligned with the scene surfaces but the dimensionality of the search space is increased from 1D (depth) to 3D (depth and two rotation angles). In this paper we present methods for aligning the correlation windows with the surfaces without exhaustive search for urban scenes. Zabulis et al. (2006) replaced the planes in plane-sweep stereo with spheres. They argue that correlation on spherical sectors along the direction of camera rays is geometrically more accurate since the surfaces where correlation takes place are always orthogonal to the viewing ray.

4.3 Plane-sweeping Stereo

In this section we outline the basic plane-sweeping algorithm. For more details we refer readers to Yang and Pollefeys (2003). Plane-sweeping stereo tests a family of plane hypotheses and records for each pixel in a reference view the best plane as scored by some dissimilarity measure. The algorithm works with any number of cameras, and images need not be rectified. The inputs to the algorithm are M planes for the depth tests, a reference image and N matching images at different camera positions (we assume images have been corrected for radial distortion), and their respective camera projection matrices P_k:

$$\mathbf{P}_k = \mathbf{K}_k[\mathbf{R}_k \quad \mathbf{t}_k] \text{ with } k = 1, \dots, N, \tag{4.1}$$

where $\mathbf{K}_k$ is the camera calibration matrix, and $\mathbf{R}_k$, $\mathbf{t}_k$ are the rotation and translation of camera $\mathbf{P}_k$ with respect to the reference camera $\mathbf{P}_{ref}$. The reference camera is assumed to be at the origin of the coordinate system. Accordingly, its projection matrix is $\mathbf{P}_{ref} = \mathbf{K}_{ref}[\ \mathbf{I}_{3\times 3} \quad 0\ ]$. The family of depth planes π_m with $m = 1, \dots, M$ is defined in the coordinate frame of the reference view by:

$$\pi_m = [\ \mathbf{n}_m^T \quad -d_m\ ] \text{ for } m = 1, \dots, M \tag{4.2}$$

where $\mathbf{n}_m$ is the unit length normal of the plane and d_m is the distance of the plane to the origin namely the center of the reference camera. (For a fronto-parallel sweep, $\mathbf{n}_m^T = [\begin{array}{ccc} 0 & 0 & 1 \end{array}]$.) The depths d_m of the planes π_m fall within the interval $[d_{near}, d_{far}]$. It is best to space the planes to account for the sampling (pixels) in the images. This is discussed in detail in Section 4.4.2.

In order to test the plane hypothesis π_m for a given pixel (x, y) in the reference view I_{ref}, the pixel is projected onto π_m and into the other images $k = 1, \ldots, N$. The mapping from the image plane of the reference camera $\mathbf{P}_{ref}$ to the image plane of the camera $\mathbf{P}_k$ is a planar mapping, and can therefore be described by the homography $\mathbf{H}_{\pi_m,\mathbf{P}_k}$ induced by the plane π_m. This homography is defined in Equation 2.9. The location (x_k, y_k) in image I_k of the mapped pixel (x, y) of the reference view is computed using homogeneous coordinates:

$$\begin{aligned} [\begin{array}{ccc} \tilde{x} & \tilde{y} & \tilde{w} \end{array}]^T &= \mathbf{H}_{\pi_m,\mathbf{P}_k} [\begin{array}{ccc} x & y & 1 \end{array}]^T \\ x_k = \tilde{x}/\tilde{w}, &\qquad y_k = \tilde{y}/\tilde{w}. \end{aligned} \tag{4.3}$$

If the plane is close to the surface projected to pixel (x, y) in the reference view, the colors of $I_k(x_k, y_k)$ and $I_{ref}(x, y)$ should be similar assuming Lambertian surfaces.

We use the sum of absolute differences (SAD) of intensities as the dissimilarity measure:

$$C(x, y, \pi_k) = \sum_{k=0}^{N-1} \sum_{(i,j) \in W} |I_{ref}(x - i, y - j) - I_k^{\star}(x - i, y - j)|, \tag{4.4}$$

where W is the matching window, and $I_k^{\star}$ is the image I_k warped by the homography $\mathbf{H}_{\pi_m,\mathbf{P}_k}$.

4.3.1 Extracting the Depthmap from the Cost

Once the cost function for all pixels and sweep planes has been computed the depthmap may be extracted. The first step is to select the best plane at each pixel in the reference view. This may simply be the plane of minimum cost, also called best-cost or winner-takes-all, defined as follows

$$\tilde{\pi}(x, y) = \underset{\pi_m}{\operatorname{argmin}}\, C(x, y, \pi_m). \tag{4.5}$$

For a given plane π_m at pixel (x, y), the depth can be computed by finding the intersection of π_m and the ray through the pixel's center. This is given by

$$z_m(x, y) = \frac{-d_m}{\begin{bmatrix} x & y & 1 \end{bmatrix} \mathbf{K}_{ref}^{-T} \mathbf{n}_m}. \tag{4.6}$$

More sophisticated approaches based on global optimization select $\tilde{\pi}$ that minimizes C but also enforces smoothness between neighboring pixels. Such methods give improved results but are too computationally expensive for real-time applications.

For occlusions, we use the solution proposed by Kang et al. (2001). For each pixel we compute the cost for each plane using the left and right subset of the cameras and select the minimum of the left and right scores. This scheme is very effective against occlusions, since typically the visibility of a pixel changes at most once in a sequence of images.

4.3.2 Implications of Cost Aggregation

To minimize C at a given pixel (x, y), the plane $\pi_m = [\mathbf{n}_m^T \quad d_m]$ should intersect not only the surface imaged at (x, y), but at all the pixels in the neighborhood window W centered at (x, y) as well. Assuming a locally planar surface, this is accomplished when $\mathbf{n}_m$ aligns with the surface normal. Misalignment of $\mathbf{n}_m$ and the surface normal potentially leads to errors in the computed depth map, depending on the size of the window, the degree of misalignment, and the surface texture (see Figure 4.2).

4.4 Multiple Sweeping Directions

We extend plane-sweeping stereo to account for non-fronto-parallel surfaces in the scene. A trivial extension would be to sample the hemisphere of all visible surface normals and sweep planes in each direction. This would lead to a large number of plane hypotheses that need to be tested, as in Zabulis et al. (2004). We propose a more efficient approach that

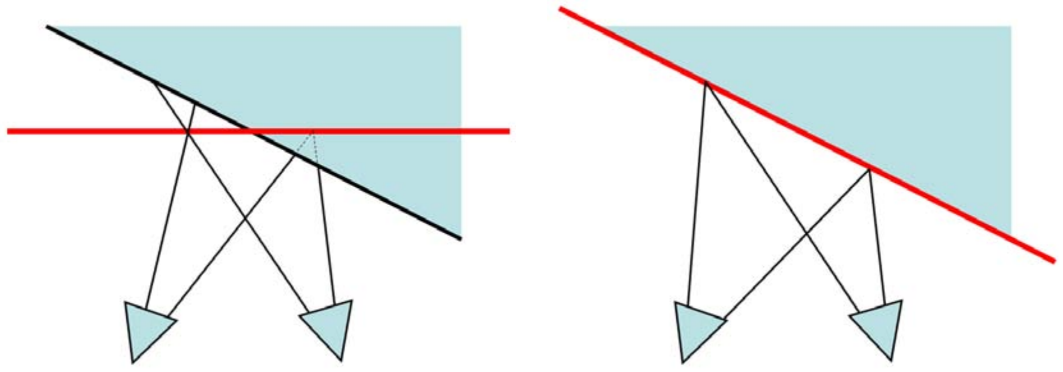


Figure 4.2: Implications of cost aggregation over a window. *Left*: Slanted surfaces with fronto-parallel plane-sweeping. Not all points over the window are in correspondence. *Right*: Surface-aligned sweeping plane to handle slanted surfaces correctly. (©2007 IEEE.)

performs multiple plane sweeps where the sweeping directions are aligned to the expected surface normals of the scene. This results in multiple depthmaps which we then combine using best-cost or, optionally, a graph cut method.

4.4.1 Identifying Sweeping Directions

Instead of exhaustively sampling the set of potential surface orientations, we can identify a much smaller set of likely surface normals either by application-specific heuristics or by examining the scene's sparse structure.

In many applications, images are captured by video or still cameras which are either hand-held or mounted on a land-based vehicle. Camera motion can be recovered either by structure from motion or from GPS/INS sensors. The motion of such cameras is generally constrained to be parallel to the ground plane, especially for vehicle-mounted, but typically also for hand-held cameras.

Additionally, a scene's planar structure can be determined by sparse features such as lines and points. This is especially true in urban environments where by examining lines in a single image, vanishing points can be recovered which in turn can be combined to give estimates of plane normals. In applications that use structure from motion, 3D point or line features are recovered as well as the camera poses. These features are available, but many algorithms do not utilize them. Many techniques have been explored to recover planar surfaces from point and line correspondences and vanishing points (Bosse et al., 2003; Hoiem et al., 2006; Schindler and Dellaert, 2004; Werner and Zisserman, 2002).

We present an effective technique for recovering planar structure in urban environments using 3D point features obtained from structure from motion. We first find the vertical direction or gravity vector. This is either given by an INS system or can be computed from vanishing points (Werner and Zisserman, 2002). Since most facades are vertical, the vanishing point corresponding to the gravity vector is quite prominent in urban scenes. By assuming the ground plane has zero slope in the direction perpendicular to the computed camera motion, we can obtain a good estimate for the ground plane normal as

$$\mathbf{G} = \frac{(\mathbf{V} \times \mathbf{M}) \times \mathbf{M}}{\|(\mathbf{V} \times \mathbf{M}) \times \mathbf{M}\|} \tag{4.7}$$

where $\mathbf{V}$ is the gravity vector, and $\mathbf{M}$ is the camera motion direction. This formula for $\mathbf{G}$ allows the ground normal to curve in the driving direction (pitch), but not roll. Thus it constrains the road to be level perpendicular to the driving direction which is typically the case. Note that obtaining the ground normal is particularly important, since the ground appears highly slanted in the image.

To compute the facade normals, we assume that they are perpendicular to the gravity vector, and are therefore determined by a rotation about the gravity vector. By assuming the facades are orthogonal to each other, only one rotation determines the facade normals. We recover the remaining rotation of the facades as follows. We first compute the orthogonal projection of each 3D point in the direction of gravity to obtain a set of 2D points. Note that 3D points on a common vertical facade will project to a line. We then evenly sample the space of in-plane rotations between 0 and 90 degrees, and then test each rotation. For each rotation $\mathbf{R} = [\mathbf{u}\ \ \mathbf{v}]^T$, we rotate the set of 2D points, and construct two histograms $H_\mathbf{u}$ and $H_\mathbf{v}$. Each bin in $H_\mathbf{u}$ (resp. $H_\mathbf{v}$) counts the number of points with a similar $\mathbf{u}$ (resp. $\mathbf{v}$) component. We then compute the entropy of each histogram, and finally select the rotation which has the lowest sum of entropies. As shown in Figure 4.3, entropy will be minimized when points are aligned in directions $\mathbf{u}$ and $\mathbf{v}$.

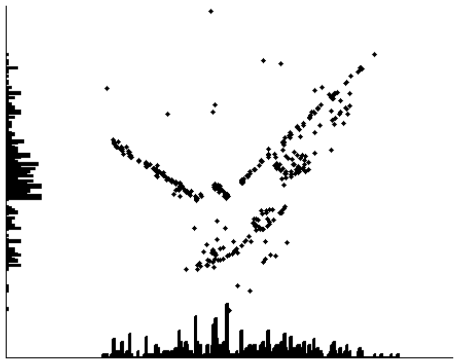
4.4.2 Plane Selection

Once the sweeping directions have been computed, we generate a family of planes for each. Referring to Equation 4.2, each family is parameterized by the distance of the plane to the origin d_m. The range $[d_{near}, d_{far}]$ can be determined either by examining the points obtained from structure from motion or by applying useful heuristics. For example, in outdoor environments, it is usually not useful for the ground plane family to extend above the camera center. The spacing of the planes in the range can be uniform, as in Zabulis et al. (2004). However, it is best to place the planes to account for image sampling (Szeliski and Scharstein, 2004). Ideally, when comparing the respective image warpings induced by consecutive planes, the amount of pixel motion should be less than or equal to one. This is particularly important when matching surfaces that exhibit high-frequency texture.

We define the disparity change between two planes $\mathbf{\Pi}_m$ and $\mathbf{\Pi}_{m+1}$ to be the maximum displacement over all pixels in all images.

$$\Delta D(\Pi_m, \Pi_{m+1}) = \max_{k=1,\ldots,N} \max_{(x,y)\in I_k} \sqrt{(x_k^m - x_k^{m+1})^2 + (y_k^m - y_k^{m+1})^2} \tag{4.8}$$

where (x_k^m, y_k^m) (resp. (x_k^{m+1}, y_k^{m+1})) are obtained by applying the homography $\mathbf{H}_{\mathbf{\Pi}_m,\mathbf{P}_k}$ (resp. $\mathbf{H}_{\mathbf{\Pi}_m,\mathbf{P}_k}$) as in Equation 4.3. To avoid orientation inversions we avoid using planes


(a) Arbitrary rotation

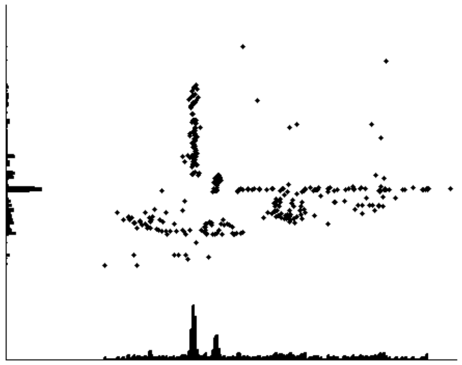

(b) Minimum entropy

Figure 4.3: Minimum entropy direction optimization. The points on the facades are obtained from structure from motion and projected in the direction of gravity. The points are then rotated into the basis formed by u and v and histograms are generated. The histograms of (a) have more entropy than those of (b). (b) corresponds to the correct surface normals. (©2007 IEEE.)

that intersect the convex hull of the camera centers. In general it is then sufficient to measure the displacements only in the cameras most distant from the reference view. Furthermore it is not necessary to compute the displacement for every pixel. The greatest displacement will occur at the boundaries of the image. Thus, to compute the disparity, we warp the polygon defined by the boundaries of image I_k into the reference view by applying the planar homography $\mathbf{H}_{\mathbf{\Pi}_m,\mathbf{P}_k}$. We then clip the polygon in the reference view, and compute the displacement of the vertices of the clipped polygon. As only planes that do not intersect the convex hull of the camera center are used, the polygon warped with $\mathbf{H}_{\mathbf{\Pi}_{m+1},\mathbf{P}_k}$ is guaranteed to remain convex, and thus the maximal disparity is bound by the maximum displacement of its vertices. The family of planes is then constructed so that the disparity change of consecutive planes is less than or equal to one pixel.

4.4.3 Incorporating Plane Priors

The minimum-entropy histograms computed in Section 4.4.1 also indicate the location of the facades. They can be used as a prior in a maximum a posteriori (MAP) formulation when selecting $\tilde{\mathbf{\Pi}}$. The posterior probability of a plane $\mathbf{\Pi}_m$ at pixel (x, y) is

$$P(\mathbf{\Pi}_m|C(x,y)) = \frac{P(C(x,y)|\mathbf{\Pi}_m)P(\mathbf{\Pi}_m)}{P(C(x,y))} \tag{4.9}$$

where $P(\mathbf{\Pi}_m)$ is the prior probability of the surface being located at plane $\mathbf{\Pi}_m$, and $P(C(x,y)|\mathbf{\Pi}_m)$ indicates the likelihood of the surface having matching cost $C(x,y)$, given that it is correctly represented by $\mathbf{\Pi}_m$. $P(C(x,y))$ is the marginal likelihood of the cost. The prior is obtained by sampling the normalized histogram at the location of the plane. For a plane $\mathbf{\Pi}_m$ chosen from sweeping direction $\mathbf{u}$, the location in the histogram $H_\mathbf{u}$ is given by the depth component of the plane d_m. The prior is

$$P(\mathbf{\Pi}_m) = \frac{H_\mathbf{u}(d_m)}{\sum_i H_\mathbf{u}(i)}. \tag{4.10}$$

The cost likelihood depends on image noise, camera pose error, alignment of the plane normal $\mathbf{n}_m$ and the surface normal, as well as on the surface texture. This is extremely difficult to model correctly. Instead we choose an exponential distribution:

$$P(C(x,y)|\mathbf{\Pi}_m) = e^{\frac{-C(x,y)}{\sigma}} \tag{4.11}$$

where σ is determined empirically. The exponential is self-similar, and so it makes no assumptions about the minimum matching cost, which is often difficult to predetermine.

Since we are only interested in the maximum likelihood solution we ignore $P(C(x,y))$ and modify the plane selection Equation 4.5 as follows:

$$\tilde{\mathbf{\Pi}}(x,y) = \operatorname*{argmax}_{\mathbf{\Pi}_m} e^{\frac{-C(x,y)}{\sigma}} P(\mathbf{\Pi}_m). \tag{4.12}$$

Maximizing this likelihood is equivalent to minimizing the negative logarithm of the likelihood. Therefore

$$\begin{aligned} \tilde{\mathbf{\Pi}}(x,y) &= \operatorname*{argmin}_{\mathbf{\Pi}_m}\{-\log e^{\frac{-C(x,y)}{\sigma}} P(\mathbf{\Pi}_m)\} \\ &= \operatorname*{argmin}_{\mathbf{\Pi}_m}\{C(x,y) - \sigma \log P(\mathbf{\Pi}_m)\}. \end{aligned} \tag{4.13}$$

Surfaces with little or no texture will exhibit a low matching cost over a range of planes, the minimum of which may be determined more by noise than by true correspondence. The prior distribution for the depth $P(\mathbf{\Pi}_m)$ helps to eliminate such ambiguities and produce a smoother surface. The implementation of Equation 4.13 comes at little additional cost and contributes significantly to the results.

We can also use the prior to significantly reduce our computation time by not testing plane hypotheses with a low prior probability. Typically a scene requires hundreds of planes for each sweeping direction to adequately sample the disparity range. While our algorithm is able to compute this many plane hypotheses at several Hz, we have found that we can obtain quality reconstructions almost an order of magnitude faster by testing only a few dozen of planes. The selected planes are those with the highest prior probability according to Equation 4.10. In our implementation, we have always chosen a fixed number of planes in order to guarantee a desired run-time. However, the number of planes could instead be adaptive to the complexity of the scene. For example, the number of planes could be chosen to capture a fixed amount of the prior's "energy", say 50%.

Note that the prior is most effective when sweeping in multiple surface-aligned directions. For example, if the sweeping direction were not aligned to the ground and were instead fronto-parallel, it would require many plane hypotheses to reconstruct the ground. However, having determined the ground's surface normal and predicted its location from the prior, we can reconstruct it with only a few plane hypotheses.

4.4.4 Selecting Depths from Multiple Sweeps

Once the plane sweeps have been performed and the best-cost solutions are selected for each sweep, the remaining task is to determine which sweeping direction at each pixel is correct. Selecting the best-cost solution already produces very good results, and most pixels are assigned the correct surface normal. However, it is evident from Figure 4.4 that due to noise some pixels are assigned incorrect normals. The immediate observation is that many of the errors are small regions of erroneous pixels embedded in a sea of correctly assigned pixels. This suggests minimizing an energy function which penalizes for abrupt changes of the surface normal within a small neighborhood. Also, the correct depths should minimize the distance between the depths of neighboring pixels. We formulate an energy which encodes the matching cost, surface normal smoothness, and depth smoothness, and minimize it using graph cuts (Boykov et al., 2001).

Typically, graph cut stereo algorithms compute the solution over a range of depth values. For the scenes in which we are interested, this would require hundreds of labels to account for the disparity range. Our energy function can be minimized efficiently since the solution is chosen from typically only a handful of sweeping directions.

Although our algorithm can be generalized to any number of sweeping directions, for simplicity the graph cut is defined in terms of three labels but can be simply extended to multiple labels. We define the set of labels $L = \{l_g, l_{f_1}, l_{f_2}\}$, with one label for each sweeping

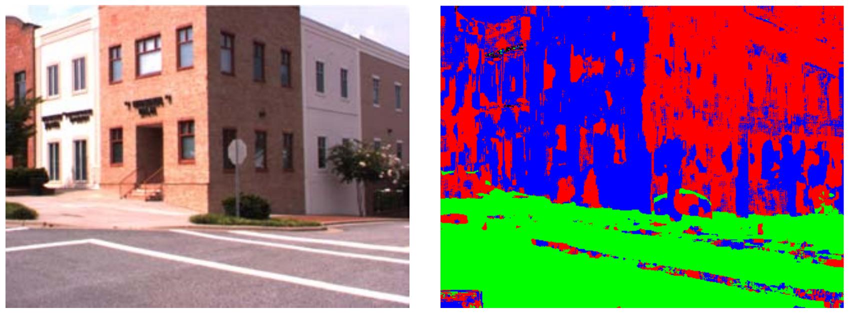


Figure 4.4: Best-cost direction selection. The labeling on the right indicates the selected sweeping direction. Ground points are labeled green, and facade points are labeled red or blue. The changes in surface normal (blue amidst a sea of red and vice versa) are clearly errors and suggest further optimization is possible. (©2007 IEEE.)

direction. Each direction also has an associated surface normal $\mathbf{n}_l$ and best cost image

$$\tilde{C}_l(x,y) = C\left(x, y, \tilde{\mathbf{\Pi}}_l(x,y)\right) - \sigma \log P(\tilde{\mathbf{\Pi}}_l(x,y)). \tag{4.14}$$

We define the energy function:

$$E = \sum_{(x,y)\in I} E_{data} + \lambda_1 \sum_{(x,y),(x',y')\in N} E_{smooth} + \lambda_2 \sum_{(x,y)\in I} E_{int} \tag{4.15}$$

where λ_1 and λ_2 adjust the relative magnitude for each penalty and $(x,y),(x',y') \in N$ indicates that (x,y) and (x',y') are 4-neighbors. The term E_{data} simply refers to the matching cost of a particular label:

$$E_{data}(x,y) = \tilde{C}_l(x,y). \tag{4.16}$$

The term E_{smooth} penalizes neighboring pixels for having different surface normals according to the Potts model (Boykov et al., 2001).

$$E_{smooth}(x,y) = \delta(l \neq l') \tag{4.17}$$

where $\delta(\cdot)$ is 1 when its argument is true and 0 otherwise.

The integrability penalty E_{int} penalizes depth discontinuities in the surface. In other stereo algorithms, this is typically defined as the absolute or squared distance between the depths of neighboring pixels. In our algorithm not only do we have the depth at each pixel,

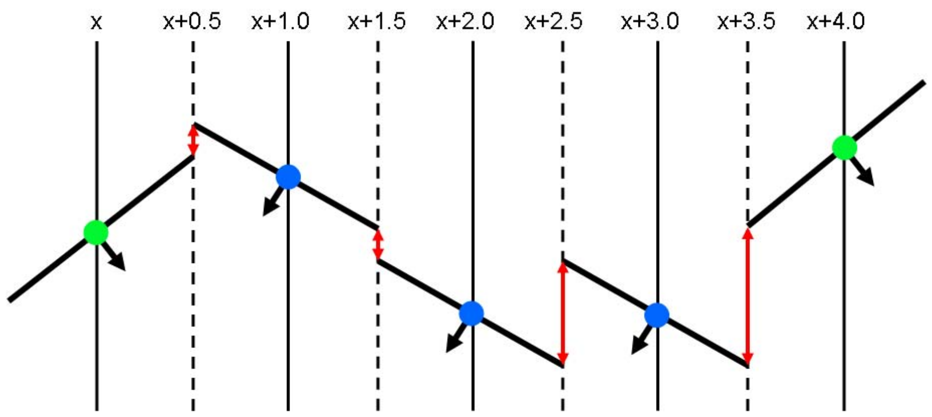


Figure 4.5: Integrability penalty for horizontal neighbors. Using the surface normals given by the labeling, the surfaces are extended to the midpoint $(x + 0.5)$. The penalty is then defined as a function of the distance between the surfaces along the line $x + 0.5$. (©2007 IEEE.)

but we have the surface normal as well. We can therefore approximate the surface at each pixel as a planar patch. For neighboring pixels, rather than compare the depths at the pixel centers, we extend each surface to the midpoint between the two pixels and compare the depths of the surfaces at this point. See Figure 4.5.

We define the integrability penalty for horizontal neighbors (x, y) and $(x + 1, y)$ with labels l and l'. (The definition for vertical neighbors is similar.) Let the plane at each pixel be $\tilde{\mathbf{\Pi}}_l$ and $\tilde{\mathbf{\Pi}}_{l'}$. We find the intersection point of each plane and the ray passing through the pixel $(x + 0.5, y)$ as in Equation 4.6. The integrability penalty is then defined as

$$E_{int}(x, y) = \min(|z_l(x + 0.5, y) - z_{l'}(x' - 0.5, y')|, E_{int}^{max}) \tag{4.18}$$

where $z_l(x + 0.5, y)$ is the intersection of the ray defined by pixel $(x + 0.5, y)$ and $\tilde{\mathbf{\Pi}}_l$, and similarly $z_{l'}(x+0.5, y)$ is the intersection with $\tilde{\mathbf{\Pi}}_{l'}$. Notationally, this assumes the pixel with label l is to the left of the pixel with label l'. The penalty saturates at value E_{int}^{max} so as not to overly penalize true discontinuities in the surface. The integrability penalty can be defined similarly for vertical neighbors. Figure 4.5 illustrates the integrability penalty. This penalty has the important property that it incurs no penalty for slanted surfaces, which do not have constant depth, as long as the depth gradient coincides with the estimated surface normal.

4.5 Experiments

We demonstrate our algorithm on video sequences captured by a hand-held camera as well as a vehicle-mounted camera as part of the Urbanscape project. The vehicle is equipped with a GPS/INS system. We compute the three sweeping directions as described in Section 4.4.1, and then compute depthmaps from 11 grayscale images with 512×384 resolution. We processed the data on an Intel Xeon 2.67 GHz PC with an NVIDIA GeForce GTX 285 graphics card.

Figure 4.6 shows an example of a depthmap computed with our algorithm from a hand-held video sequence. (A 3D model of this scene is shown in Figure 4.10). A close-up of the depthmap computed with a fronto-parallel sweep reveals the staircase artifact on the slanted ground surface. Because only part of the matching window is in correct correspondence for the correct plane, other parts of the window may dominate the matching score especially if there is stronger texture than at the center pixel. Thus the nearby pixels will "lock on" to the highest nearby frequency, producing the artifact. By aligning the sweeping plane to the surface, the entire window is in good correspondence and the artifact is eliminated.

Rather than present depthmaps, we will use 3D models to present the remaining results. As discussed in Chapter 2, obtaining 3D models from depth measurements is a non-trivial problem which is addressed specifically in Chapter 7. For the present results, surfaces are defined by pixel adjacency except where a depth discontinuity is detected by thresholding.

In the next experiment, we compare our algorithm with the basic fronto-parallel sweep. The scene is a flat brick wall which is viewed obliquely and was captured by the Urbanscape system. We reconstruct the scene using 144 plane hypotheses for both algorithms. For our algorithm we selected the depths from the multiple sweeping directions using best-cost, and achieved a processing rate of 11.76 Hz. (The graph-cut method was not used for this experiment.)

Figure 4.7 compares a scanline of the depthmaps resulting from the fronto-parallel sweep algorithm and our algorithm. The surface from our algorithm is much smoother and better approximates the planar facade. Note that for both algorithms we compute sub-pixel matches by parabolic fitting of the best cost. Despite this fact, the fronto-parallel sweep is

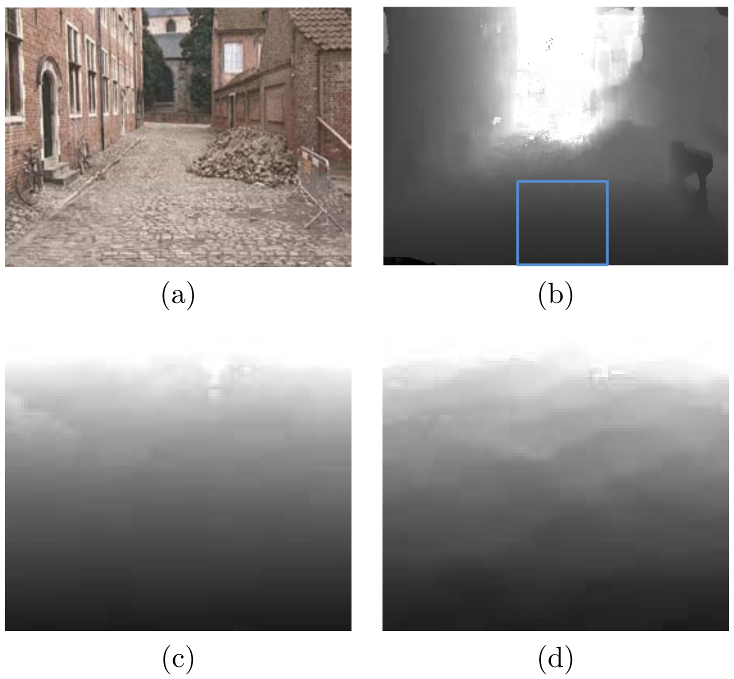


Figure 4.6: (a) Original image. (b) Depthmap computed using our multi-sweep method. Close-ups are shown for the marked area. (c) Close-up of the multi-sweep depthmap. (d) Close-up of the fronto-parallel sweep depthmap. Notice the staircase artifacts.

unable to compute a smooth surface. Since only a percentage of the points in the aggregation window actually have the correct depth, the matching costs to which the parabola is fit are less indicative of the depth of the surface.

The following scene is also captured by a vehicle-mounted camera from Urbanscape. In this scene, shown in Figure 4.8, we demonstrate the graph-cut-based depth selection from Section 4.4.4. Selecting the best-cost sweeping directions already produces a surface normal labeling which is quite accurate. However, small errors remain. The graph cut is able to eliminate the incorrectly labeled regions and straighten out the interface between the two facades. The cross-sections in Figure 4.8 shows that by selecting the correct surface normals labels the computed depthmap is improved. This graph cut, although unsuitable for real-time, is quite efficient and takes less than 2 seconds to compute. As a follow-up to this approach, Zach et al. (2008) developed a variational method for solving the 3-label energy. Unlike graph-cuts, this method runs on the GPU, but it still takes 45 milliseconds to compute which is too slow for real-time.

In Figure 4.9, we demonstrate the ability to compute accurate depthmaps with only a

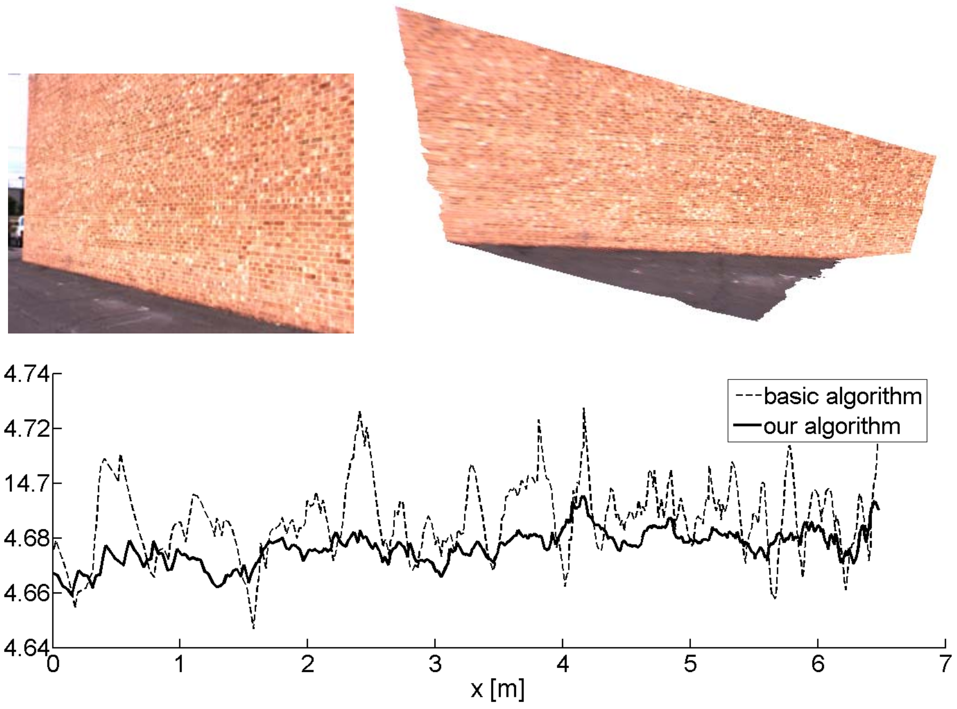


Figure 4.7: Comparison of the basic fronto-parallel plane sweep and sweeping in multiple directions. *Top left*: the original viewpoint. *Top right*: The depthmap triangulated into a polygonal mesh viewed from above. *Bottom*: A cross-section of the surface. Note the scale on the axes. We measured the standard deviation of the surface from the best-fit line. This was 1.31 cm for the fronto-parallel sweep and 0.61 cm for our algorithm. (©2007 IEEE.)

small number of plane hypotheses. By testing only the planes with highest prior probability, we produced quality reconstructions with just 48 plane hypotheses per frame. This increases the speed of our algorithm to 50.0 Hz. Although we assume the presence of planes in the scene, our algorithm is a general stereo matcher (see Section 4.3), and we are still able to reconstruct non-planar objects such as the bushes, even with only 48 distinct planes.

Finally, we present the reconstruction of a scene captured by a hand-held camera. We computed the direction of the gravity vector from vertical vanishing points and computed the sweeping directions as described in Section 4.4.1. The reconstruction of the scene is shown in Figure 4.10. By rendering the scene with a synthetic light source we can see that without aligning the sweeping plane to the surface normals, the reconstructed surface is quite bumpy. Our algorithm produces a reconstruction which is much smoother.

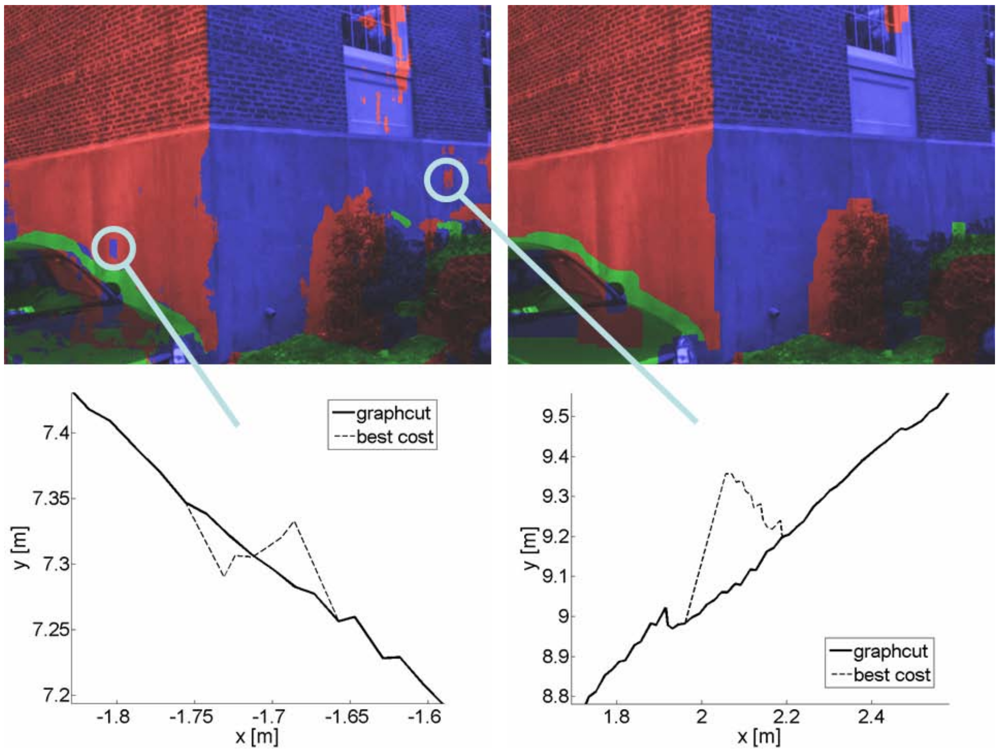


Figure 4.8: *Top*: The best-cost sweeping direction labeling (left) and the labeling after the graph cut (right) Several mislabeled pixels are corrected by the graph cut. *Bottom*: The improvement in accuracy after the graph cut. This shows the incorrectly labeled pixels were indeed errors in depth. (©2007 IEEE.)

4.6 Discussion

In this chapter, I have presented a real-time multi-view stereo algorithm based on plane-sweeping which correctly handles slanted surfaces. A real-time plane-sweeping approach is needed to process the massive amounts of video data required to reconstruct at city-scale. The algorithm runs on the GPU at up to 50 Hz on 512×384 video. To better handle slanted surfaces, which are a problem for real-time stereo methods, we propose a multi-sweep approach. An estimate of the normals of the planar surfaces in the urban scene is obtained by analyzing sparse point correspondences, which are available from the structure from motion estimation. Urban scene properties such as verticality and orthogonality allow the building facade surface normals to be obtained by an efficient alignment procedure. Plane-sweeps are then performed along these directions and results are combined according to best-cost or, optionally, a simple three-label graph cut. Furthermore, by incorporating

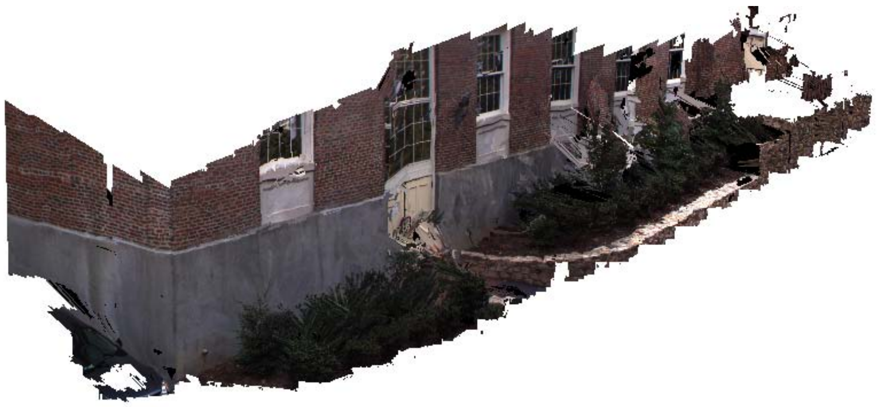
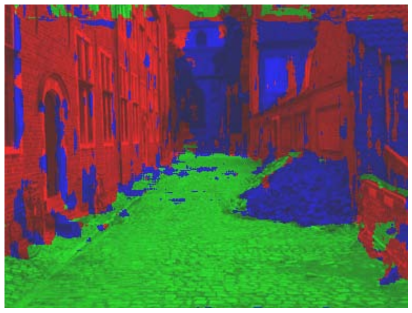
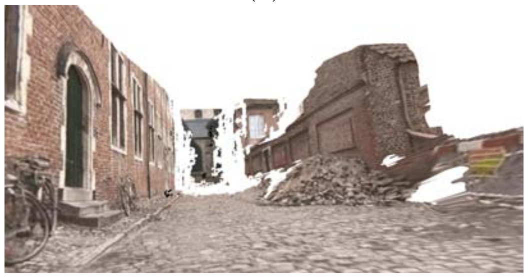


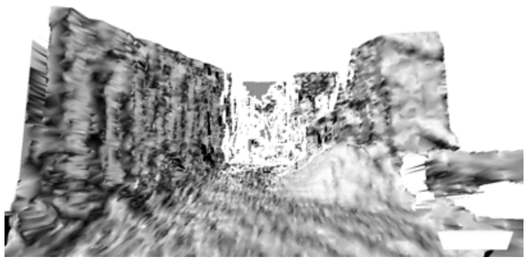
Figure 4.9: The reconstruction produced by our algorithm from several hundred frames of video. By testing only plane hypotheses with high prior probability, the reconstruction was achieved with only 48 plane hypotheses per frame at a speed of 50.0 Hz.

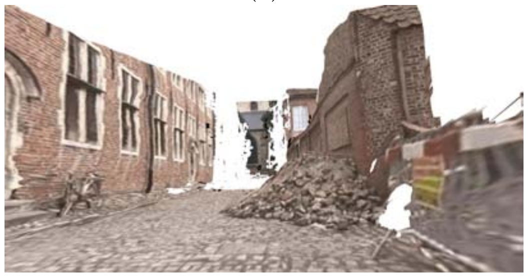
plane priors given by the sparse point correspondences, we can reduce computation time and improve results in ambiguous parts of the scene. Comparisons with ground truth show a clear improvement over the basic fronto-parallel plane-sweeping algorithm. Thus this algorithm can be seen as a method to reduce the correspondence error in Equation 2.11.

(a)


(b)


(c)


(d)


(e)

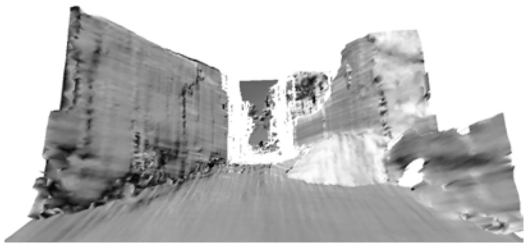

(f)

Figure 4.10: *(a)*: Frame of video captured by a hand-held camera. *(b)*: The best-cost labeling of sweeping direction. *(c) and (e)*: Novel views of the reconstructed scene. *(d) and (f)*: The reconstructed scene rendered as a polygonal mesh with a light source. (d) is from the basic plane sweeping algorithm, and (f) is from our algorithm. The lighting demonstrates a much smoother surface is obtained by aligning the sweeping plane to the surface normals. (©2007 IEEE.)

CHAPTER 5

Variable Baseline/Resolution Stereo

5.1 Introduction

In the previous chapter, more accurate correspondences were obtained by explicitly accounting for slanted surfaces in stereo. In this chapter, the focus will be on determining the uncertainty of the obtained depth measurements, and selecting views and working resolution to balance that uncertainty with computational effort.

Recent state-of-the-art stereo methods have been very successful in solving the correspondence problem, which is to decide which pixels in one image correspond to which pixels in another. Techniques employing graph cuts and belief propagation can achieve error rates of less than 1% on laboratory data (Scharstein and Szeliski, 2002). However, for many applications, including urban 3D reconstruction, the goal is ultimately not pixel correspondence but depth accuracy. Even with pixel-accurate correspondences, the depth error in traditional stereo grows *quadratically* with depth (Equation 2.11), which means that the accuracy in the near range far exceeds that of the far range. While the accuracy in the far range is unusably bad, the accuracy in the near range is unnecessarily high and comes at significant computational cost. Accuracy can be improved by incorporating multiple views. These views provide additional information which aids in the correspondence problem, but they can also improve the depth accuracy *geometrically* by increasing the angle of triangulation. In many applications, such as structure from motion from video (Pollefeys et al., 2008), or recently reconstruction from community photo collections (Goesele et al., 2007; Furukawa et al., 2010), the choice of views for stereo is quite flexible. Our technique focuses on selecting the best cameras, as well as the most appropriate sampling in the images, to compute a depthmap that meets the desired geometric accuracy with minimal computation.

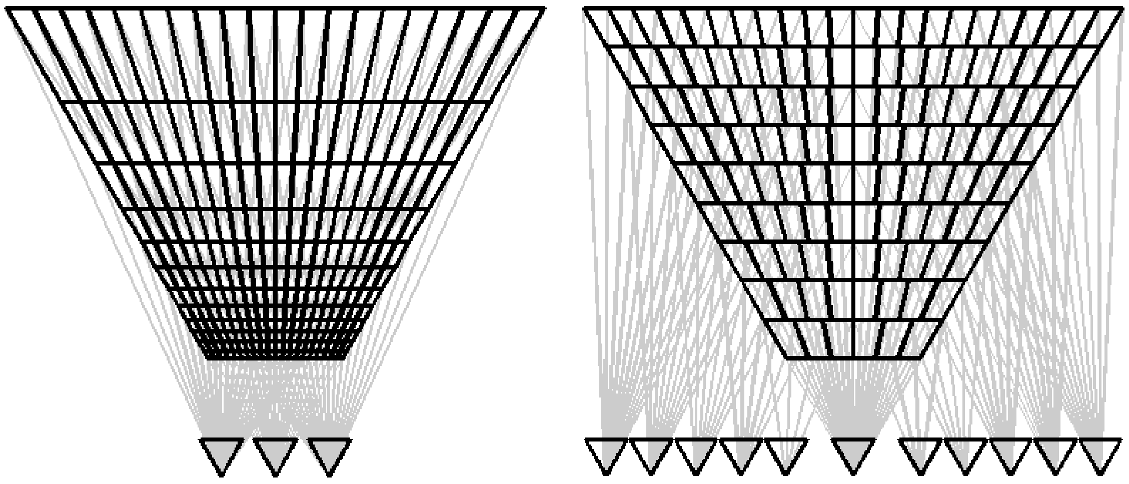


Figure 5.1: *Left*: Standard stereo. Note that the distance between depths increases quadratically. *Right*: Variable Baseline/Resolution Stereo. The distance between depths is held constant by increasing the baseline and selecting the appropriate resolution. (©2008 IEEE.)

Specifically, we increase the baseline to increase accuracy in the far range, and we reduce the resolution (using a gaussian pyramid) to reduce computational effort in the near range.

This approach is important for large-scale urban reconstruction, because scenes captured from the ground exhibit a large depth range, i.e. objects can be observed both near and far from the camera. Managing depth uncertainty over this large range is not only important for obtaining accurate depth measurements in the far range, but it is also important for reducing computational effort in order to efficiently process large amounts of video.

The motivation for our approach derives from the point of view of a system designer wishing to employ stereo as a measuring device. Often, the definition of the stereo problem assumes the camera parameters are given and fixed, but these parameters have a significant effect on the depth accuracy of stereo. The system designer has some accuracy requirements in mind and, with traditional stereo methods, must carefully select baseline, focal length, and field of view in order to meet these requirements. Furthermore, computation time is important in real systems, and so the designer must be conservative. It is unacceptable to spend large amounts of time obtaining accuracy that far exceeds the minimum requirement. Balancing accuracy and efficiency for standard stereo is difficult indeed due to its quadratic error characteristics. Our novel algorithm enhances stereo to be able to efficiently use the additional information contained in dense image sets such as video by dynamically selecting the appropriate baseline and image scale for each depth estimate. In contrast to traditional stereo our technique guarantees a constant target accuracy throughout the greatest possible volume with orders of magnitude less computational effort.

5.1.1 Variable Baseline/Resolution Stereo

The motivation for our algorithm is derived from the depth error in stereo, which in Equation 2.11 was shown to be

$$\epsilon_z = \frac{z^2}{bf} \cdot \epsilon_d \tag{5.1}$$

where ϵ_z is the depth error, z is the depth, b is the baseline, f is the focal length of the camera in pixels, and ϵ_d is the correspondence error in pixels (disparity values). The focus of this algorithm will be on the *geometric factor* of the depth error, which is $z^2/(bf)$. This quantity also determines how the resolution of the image translates into the resolution of depth measurements in space. Since disparities are measured in pixels (spaced 1 unit apart), the spacing between depth measurements, which we will call *depth resolution*, is also $z^2/(bf)$. The depth resolution can be controlled by varying the baseline and focal length. Dense image sets, such as video, allow the baseline b to be selected with great flexibility, and because f is measured in pixels, the focal length can be varied by selecting the appropriate scale in a Gaussian pyramid (up to the maximum value of f at full image resolution). The principal idea of our algorithm is to set both b and f proportionally to z throughout the depthmap computation, thereby canceling the quadratic z term, and leaving the depth resolution *constant* w.r.t. depth. Thus matching scores for depths in the near range are computed using a narrow baseline and coarse image resolution, while depths in the far range use a wider baseline and finer image resolution.

While the focus of our approach is the geometric factor, multiple views as in video can have a beneficial effect on correspondence error. Averaging (unbiased) measurements over multiple views reduces uncertainty. The mean of n uncorrelated measurements with variance σ^2 is σ^2/n, so the uncertainty (standard deviation) improves with $\sqrt{n}$. But depth measurements from video are often highly correlated. For example, Geyer et al. (2006)

found a correlation coefficient of 0.6. In our implementation, we do use multiple views for matching, but it is a fixed number, regardless of baseline. Using a larger number of views for larger baselines would help to reduce correspondence error, but in practice we have not found a significant improvement by using more images. Also there are many factors like texture which influence the correspondence error besides the number of views. While we acknowledge that correspondence error can be improved somewhat with more views over larger baselines, the focus of our algorithm is the geometric factor that scales the correspondence error.

Our algorithm, which we call Variable Baseline/Resolution Stereo, exhibits three important properties:

1. By selecting the baseline and resolution proportionally to the depth, we can match the quadratic term in the depth error, and achieve constant depth resolution over the reconstructed volume.

2. Because the depth resolution is constant throughout the reconstructed volume, the computational effort is also evenly spread throughout the volume.

3. The baseline grows linearly with depth, therefore the angle of triangulation remains constant[1].

To the best of our knowledge, our method is the first to exhibit all three properties.

In the following sections I will consider previous work relevant to this chapter, analyze the error and time complexity of our algorithm as compared to traditional stereo, discuss the implementation of our algorithm, and present results.

This work in this chapter was published in Gallup et al. (2008).

5.2 Related Work

While, to the best of our knowledge, no prior work on stereo with both variable baseline and resolution has been published, there is a significant amount of research on each of

[1]The angle of triangulation is constant w.r.t. to depth. It varies slightly from pixel to pixel.

these aspects separately. In this section, we briefly review some papers that deal with multi-baseline or multi-resolution stereo. We also review reconstruction approaches that explicitly model geometric uncertainty.

Early research on multi-baseline stereo includes the work of Okutomi and Kanade (Okutomi and Kanade, 1993) who use both narrow and wide baselines, which offer different advantages, from a set of cameras placed on a straight line with parallel optical axes. However the goal was to reduce mismatches, not to improve depth accuracy. Sato et al. (2002) address video-based 3D reconstruction using hundreds of frames for each depth map computation. The median SSD between the reference and all target views is used for robustness against occlusions. In general, using multiple images improves matching, and employing wider baselines can increase the depth resolution. However, none of these approaches guarantee bounds on depth resolution. Our approach is unique in that we use a different set of images throughout the computation such that the baseline grows proportionally to depth.

Multi-resolution approaches are used for stereo to either speed up computation or to combine the typically less ambiguous detection at coarse resolution with the higher precision of fine resolution. The latter was the motivation for the approach of Falkenhagen (1997) in which disparities are propagated and refined as processing moves from coarse to fine levels of image pyramids. Yang and Pollefeys (2005) presented an algorithm in which cost functions from several different resolutions were blended to take advantage of the reduced ambiguity coming from matching at coarse levels of the image pyramids and the increased precision coming from matching at fine levels. Koch et al. (1998) use a multi-resolution stereo algorithm to approximately detect the surfaces quickly, since processing speed is important for large scale reconstruction systems which operate on large disparity ranges. Reducing image resolution results in an equivalent reduction of the disparity range. Sun (2002) presented a method that aims at improving both the speed and reliability of stereo. It operates in bottom-up fashion on an image pyramid in which stripes are adaptively merged to form rectangular regions based on disparity similarity. A two-stage dynamic programming optimization stage produces the final depth map. In these approaches, multiple resolutions are used for speed and/or improved matching, but depth accuracy is not addressed. A key component of our algorithm is that we use different resolutions at different depths. Specifically, we use lower resolutions to estimate depths in the near range in order to avoid unnecessary computations for resolution that far exceeds what is required.

Algorithms that take geometric uncertainty explicitly into account include Matthies et al. (1989) and Koch et al. (1998). Matthies et al. introduced an approach based on Kalman filtering that estimates depth and depth uncertainty for each pixel using monocular video inputs. These estimates are refined incrementally as more frames become available. Koch et al. proposed a similar approach that computes depth maps using pairs of consecutive images. Support for each correspondence in the depth maps is found by searching adjacent depth maps both forward and backward in the sequence. When a match is consistent with a new camera, the camera is added to the chain that supports the match. The position of the reconstructed 3D point is updated using the wider baseline. While these methods are successful in reducing error in the reconstruction, they do not exhibit the properties from Section 5.1.1. In particular, the computational effort is concentrated in the near range, and as a result the depth accuracy in the near range exceeds that of the far range.

5.3 Analysis

Before analyzing the accuracy and time complexity of our stereo algorithm, we shall briefly address the issue of depth sampling. As outlined in the simple example in Algorithm 2.2, stereo seeks to determine the depth of the surface along the rays passing through each pixel in a reference image. Each point along the ray is projected into any number of target images, and a measure of photoconsistency is computed. This defines a function of depth, the minimum (or maximum) of which indicates the depth of the surface and is discovered by sampling the function at various depths. The number and location of the samples should be defined by the pixels in the target images (disparities). While supersampling can obtain a more accurate minimum, the depth resolution is still proportional to the pixels. Szeliski and Scharstein (2004) observed that sub-pixel accuracy up to 1/4 pixel is typical. Note also that subsampling the function without properly filtering the images will lead to aliasing, and the minimum of the aliased function can often be far from the minimum of the original function. Therefore the sampling rate should be on the order of one pixel. In stereo, one cannot expect to obtain greater depth accuracy simply by finer disparity sampling, and in order to use coarser sampling (to reduce computation time and accuracy), filtered lower-resolution images must be used. For more details on sampling in stereo, see (Szeliski and Scharstein, 2004).

5.3.1 Accuracy and Time Complexity

We now analyze the time complexity of traditional stereo and compare it to the time complexity of our variable baseline/resolution algorithm. Our analysis assumes that the cameras are separated by lateral translation and no rotation, so that all cameras share a common image plane, and pixel correspondences have the same vertical image coordinate. This setup, which is convenient for analysis, can be somewhat relaxed in the actual implementation of our algorithm.

In our analysis we assume that the system designer specifies a desired accuracy: a minimum depth resolution Δz, and a maximum range z_{far}. Assuming correspondence error is sub-pixel, stereo is expected to deliver depth measurements with error less than Δz for all depths $z \leq z_{far}$. The depth resolution is defined similarly to the depth error, replacing ϵ_d with the pixel spacing, equal to 1:

$$\Delta z = \frac{z^2}{bf}. \tag{5.2}$$

Therefore meeting the target depth resolution depends on adjusting the baseline b and focal length f.

We will now analyze the effect of baseline and focal length separately, then combined, followed by an analysis of our algorithm. We focus our analysis on the target accuracy parameters Δz and z_{far}.

Fixed-baseline stereo. For a fixed-baseline stereo system, the depth resolution can be adjusted by varying the focal length parameter f. Since f here is measured in pixels, it can be increased either by narrowing the field of view (zoom), or by increasing the resolution of the image sensor. We assume the field of view θ_{fov} has been carefully chosen for the application, meaning f describes the image resolution as

$$w = 2f \tan \frac{\theta_{fov}}{2} \qquad h = \frac{w}{a}, \tag{5.3}$$

where w, h and a are the width, height and aspect ratio of the image. We can determine the image resolution needed to meet the target accuracy by solving for f in Equation 2.11.

$$\begin{aligned} f &= \frac{{z_{far}}^2}{b\Delta z} && (5.4) \\ \textit{number of pixels} &= wh = \frac{w^2}{a} \\ &= \frac{{z_{far}}^4}{\Delta z^2} \frac{4\tan^2 \frac{\theta_{fov}}{2}}{b^2 a} && (5.5) \end{aligned}$$

This shows that increasing the image resolution alone to meet the target accuracy requires the resolution to grow proportionally to ${z_{far}}^4$! Note that a higher resolution sensor does not necessarily increase the effective resolution. Higher quality lens optics may also be required, making it prohibitively expensive, or impossible, to increase the resolution at this rate. Another prohibitive factor is the processing time. In stereo, each pixel must be tested against the pixels along the corresponding epipolar line within the disparity range of the scene. Because the *depth* range is defined by the scene, the *disparity* range is some fraction of the image width, and thus also increases with image resolution. Letting D be the ratio of the disparity range to the image width, the number of pixel comparisons needed is

$$\begin{aligned} T_{\textit{fixed}} &= Dw^2h = \frac{Dw^3}{a} \\ &= \frac{{z_{far}}^6}{\Delta z^3} \frac{8D\tan^3 \frac{\theta_{fov}}{2}}{b^3 a} \\ &= O({z_{far}}^6 \Delta z^{-3}). && (5.6) \end{aligned}$$

This means the system designer is severely limited by depth range. For example, extending the depth range by a factor of 2 would require $2^6 = 64$ times more computational effort!

Fixed-resolution stereo. If the image resolution is held fixed, the depth resolution can be increased by increasing the baseline b. To meet the target accuracy, we solve Equation 2.11 for b, yielding $b = \frac{{z_{far}}^2}{f\Delta z}$. One drawback of increasing the baseline is that the depth where the fields of view begin to overlap also increases, and the near range is lost. The depth where the overlap begins is $z_{near} = \frac{b}{\tan\theta_{fov}/2}$. Because z_{near} depends on b, and b grows quadratically with z_{far}, there is a point at which z_{near} surpasses z_{far}, meaning that the depth where the target accuracy is met is no longer in the overlapping field of view. In general, one cannot rely on increasing the baseline alone to meet the target accuracy.

Variable baseline and resolution. In order to avoid z_{near} surpassing z_{far}, the baseline cannot grow faster than linearly with z_{far}. Thus we set $b = \beta z_{far}$ where β can be chosen to give a certain angle of triangulation at z_{far}. Given this constraint, we solve for the image resolution needed to meet the target accuracy as follows:

$$\begin{aligned} f &= \frac{{z_{far}}^2}{b\Delta z} = \frac{z_{far}}{\beta\Delta z} && (5.7) \\ \textit{number of pixels} &= \frac{{z_{far}}^2}{\Delta z^2}\frac{4\tan^2\frac{\theta_{fov}}{2}}{\beta^2 a}. && (5.8) \end{aligned}$$

From this equation we see that the baseline and the focal length both grow linearly with z_{far}, and the required depth resolution grows proportionally with ${z_{far}}^2$ rather than ${z_{far}}^4$. However, with a linearly growing baseline, z_{near} also grows linearly, and overlap in the near range is lost. Therefore, in order to accurately reconstruct the entire scene, wide baselines must be used in the far range, and narrow baselines must be used in the near range.

We now analyze our method which uses multiple baselines and resolutions to recover depths over the entire viewing volume with minimal computational effort. Unlike previous approaches which combine measurements among multiple baselines and resolutions, our method chooses a single baseline and resolution based on the depth being measured. This approach has several advantages mentioned in Section 5.1.1: 1) the depth error is constant for all depths, 2) the amount of computational effort is evenly distributed throughout the volume, 3) the angle of triangulation does not vary with depth.

For the sake of analysis, assume that the stereo setup consists of a continuous set of cameras with baselines given by the function $\mathcal{B}(x) = xb, 0 \leq x \leq 1$, where b is the required baseline from Equation 5.7. For each of these cameras there is an image I_x which has been constructed as a scale pyramid, again, with a continuous set of scales. The focal length (in pixels) of the scales is given by the function $\mathcal{F}(x) = xf, 0 \leq x \leq 1$, where f is the required focal length from Equation 5.7. In reality, baselines and scale pyramid levels are discrete; however, the set of baselines acquired from a moving camera is quite dense, and the continuous scale pyramid can be approximated by filtering between the two nearest discrete levels.

Since our method uses multiple images, it is more convenient to parameterize correspondences in terms of their triangulated depth z instead of their pixel coordinate disparity d. Our approach varies the baseline and resolution with z. The baseline is chosen as $\mathcal{B}(z/z_{far})$, and the resolution is chosen such that the focal length is $\mathcal{F}(z/z_{far})$. By substituting this baseline and focal length into the depth resolution Equation 5.2, we see that the depth resolution is equal to our target, Δz for all $z \leq z_{far}$.

To analyze the time complexity of our algorithm, we sum the number of pixel comparisons needed at each depth z. Since we step through depth at a constant rate Δz, there are $z_{far}/\Delta z$ steps. Letting $k\Delta z/z_{far}$ be the proportion of the width and height required at depth $k\Delta z$, the time complexity can be expressed as a sum:

$$\begin{aligned} T_{variable} &= \sum_{k=1}^{\frac{z_{far}}{\epsilon_z}} wh \left(\frac{k\epsilon_z}{z_{far}}\right)^2 = \frac{wh\Delta z^2}{{z_{far}}^2} \sum_{k=1}^{\frac{z_{far}}{\Delta z}} k^2 \\ &= \frac{4\tan^2 \frac{\theta_{fov}}{2}}{a\beta^2} \left(\frac{{z_{far}}^3}{3\Delta z^3} + \frac{{z_{far}}^2}{2\Delta z^2} + \frac{z_{far}}{6\Delta z}\right) \\ &= O({z_{far}}^3 \Delta z^{-3}). \end{aligned} \tag{5.9}$$

This is a considerable improvement over standard stereo which is $O({z_{far}}^6 \Delta z^{-3})$ as shown in Equation 5.6. Note that the reconstructed volume is a frustum (pyramid) ranging from the camera to z_{far}. If we divide this volume into voxels with side length Δz, the number of voxels in the volume is also $O({z_{far}}^3 \Delta z^{-3})$. Without prior knowledge, each voxel must be visited, or at least a number of voxels proportional to the volume must be visited, to reconstruct the volume. Under these assumptions, $\Omega({z_{far}}^3 \Delta z^{-3})$ is the asymptotic lower bound for stereo, which our algorithm achieves.

While our analysis has focused on image-centered stereo, we briefly mention a different class of stereo, namely volumetric methods (Dyer, 2001; Kutulakos and Seitz, 2000). By nature, the time complexity of volumetric stereo is proportional to the volume, and the computation time is spread evenly over the volume (property 2 from Section 5.1.1). However, these methods do not explicitly guarantee the target accuracy. In order to do so, voxel size and camera selection must be chosen such that the projection of each voxel differs from the projection of the neighboring voxels by exactly one pixel in some camera. Assuming pixel accurate matching, this ensures that each voxel is visually distinguishable from its neighbors, and therefore the surface can be located to within one voxel in space. To the best of our knowledge, no volumetric method exists which guarantees uniform depth resolution over the entire volume.

5.4 Algorithm

We use a plane-sweeping approach (see Chapter 4) to compute a depthmap for a reference view using multiple target views. A key difference from the algorithm presented in Section 4.3, is that the matching views can be different for each plane. In our algorithm, for each depth plane z, we choose a constant number of images whose baselines are evenly spread between $-\mathcal{B}(\frac{z}{z_{far}})$ and $\mathcal{B}(\frac{z}{z_{far}})$. Finally, to ensure low correspondence error, and to focus attention on depth resolution, we have used semi-global optimization (Hirschmuller, 2008) which is a good balance between efficient local methods and accurate global methods. This replaces the best-cost approach used in Section 4.3.1.

In the actual implementation of our algorithm, the set of cameras and their associated image pyramids are finite and discrete, and we do not require that cameras be strictly constrained to lateral translation and no rotation. Thus we can only approximate the accuracy and pixel motion by the simple formulas previously mentioned. Instead we measure the image sampling and accuracy directly by projecting the hypothesized depths into the leftmost and rightmost views. As discussed in Section 4.4.2, we only need to measure at the vertices of the projected and clipped image border polygon, which bounds the measurements of all interior points.

Algorithm 2 Variable Baseline/Resolution plane-sweep. The baseline and resolution increase from narrow to wide and from coarse to fine as necessary to maintain the target error bound.

```
$z \Leftarrow z_{near}$
$b' \Leftarrow$ narrowest baseline
while $z \leq z_{far}$ do
  compute matching scores for depth plane $z$
      and store in cost volume
  choose pyramid level $f' = \mathcal{F}(z/z_{far})$
  while $z^2/(b'f') > \Delta z$ do
    increase baseline $b'$
  end while
  $z \Leftarrow z + \Delta z$
end while
compute depthmap from cost volume
```

Algorithm 4.2 describes our method. The plane-sweep begins with the narrowest baseline and coarsest resolution. As the sweep moves from near to far, the baseline and resolution (pyramid scale) increase from narrow to wide and from coarse to fine as necessary to maintain the target accuracy. Once the full image resolution is attained, the plane-sweep can continue with increasing baselines, but the depth resolution bound will no longer be met. Matching scores are computed at each depth and stored in a cost volume, from which an optimized surface is then extracted.

In our method we compare matching costs computed at different baselines and resolutions and expect that the minimum score indicates the correct match. For multiple baselines, we expect this to be true based on the brightness constancy constraint. Although appearance changes are a known problem in wide baseline matching, our method is *not* wide baseline since the angle of triangulation is kept approximately constant and relatively small. In Figure 5.2, we show an example of matching costs computed from various resolutions and baselines. This figure shows that the cost minimum is roughly the same for all cost functions. We have evaluated this for a variety of scenes and pixels and found the same general behavior in all of them.

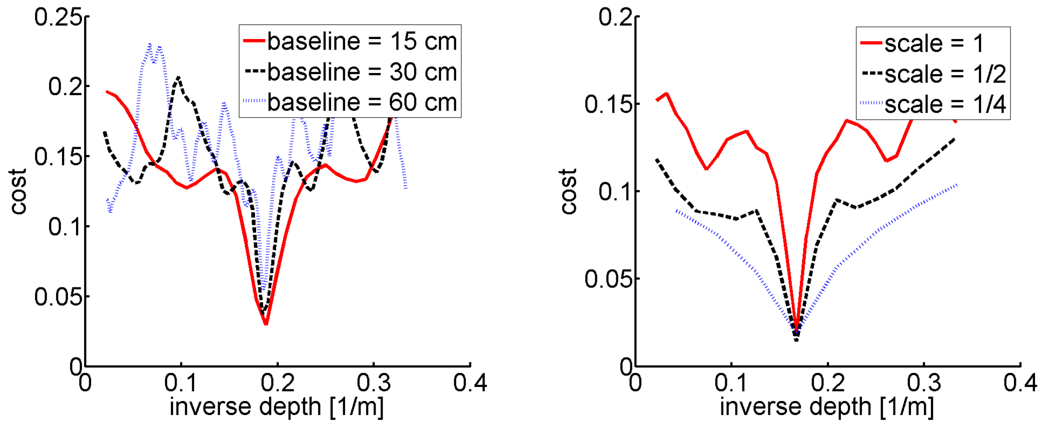


Figure 5.2: Matching cost functions for varying baselines (left) and resolutions(right). Cost minima are roughly the same value at different resolutions and baselines, which makes stereo matching possible across different resolutions and baselines. (©2008 IEEE.)

5.5 Experiments

We have evaluated our method and standard stereo against a scene for which ground truth was acquired with a laser range finder. The scene, shown in Figure 5.3, is simple by design, so that the focus is depth accuracy, not matching accuracy. The scene features a slanted brick wall which ranges from 7 to 12 meters in depth. For our method, we set $\epsilon_z = 10cm$ and $\beta = \tan 10°$ (i.e. 10° angle of triangulation). To evaluate error w.r.t. depth, we divided the computed depths into bins spaced $25cm$ apart, and computed the standard deviation of the (signed) difference from ground truth. As expected, the error in standard stereo grows with depth, whereas the error from our method remains constant. Note that our target accuracy is $10cm$, whereas the average standard deviation is $5cm$, from which we can deduce the standard deviation of the correspondence error ϵ_d is 0.5 pixels (see Equation 2.11).

Next we evaluated our algorithm on challenging outdoor scenes. The first scene was acquired with a 1024x768 pixel video camera undergoing lateral motion and capturing images at 30 Hz. The field of view was 40 degrees. For this scene, we desired an accuracy of $\Delta_z = 30cm$, and have found matching to be accurate at angles up to 6 degrees, i.e. . $\beta = \tan 6°$. Given our resolution, the target accuracy can be maintained up to $45m$. We used a gaussian pyramid where the scaling factor between levels is 1/2, and filtered between

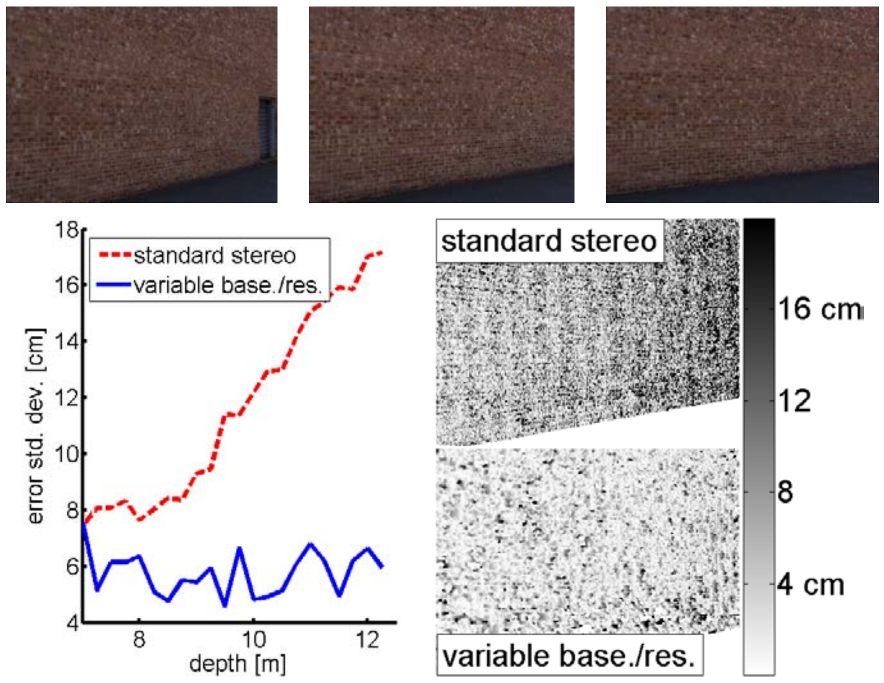


Figure 5.3: We compared standard stereo and our algorithm against a scene for which ground truth was acquired with a laser range finder. The depth of the wall ranges from 7 to 12 meters. *Top*: Some original images. *Bottom Left*: Standard deviation from ground truth w.r.t. depth. The error of standard stereo increases with depth while the error of our algorithm remains roughly constant. *Bottom Right*: Absolute error images (darker means larger error). (©2008 IEEE.)

the two nearest levels to handle variable resolution. Depthmaps were computed using the previously described plane-sweep, using 11 views, followed by semi-global optimization. We have compared our results with standard stereo, also using 11 views. For a fair comparison, we allowed standard stereo to use the widest baseline possible, while still keeping the objects in the near range in view. The near range in this scene is $3m$ which limits the baseline to $2.5m$. This baseline is in fact not sufficient to meet the target accuracy at the far range. Except for the differences in baseline and resolution, all other settings are the same for both methods. The two methods are compared in Figure 5.4. Our method is more than 6 times faster, and analysis predicts that it is more than 4 times more accurate at the far range. While no ground truth is available for this scene, the reconstruction produced using our method is clearly many times more accurate.

For the second result, shown in Figure 5.5, we captured a series of images with a 10 megapixel camera. We processed the images with a target accuracy of $\Delta_z = 1cm$ at $z_{far} =$

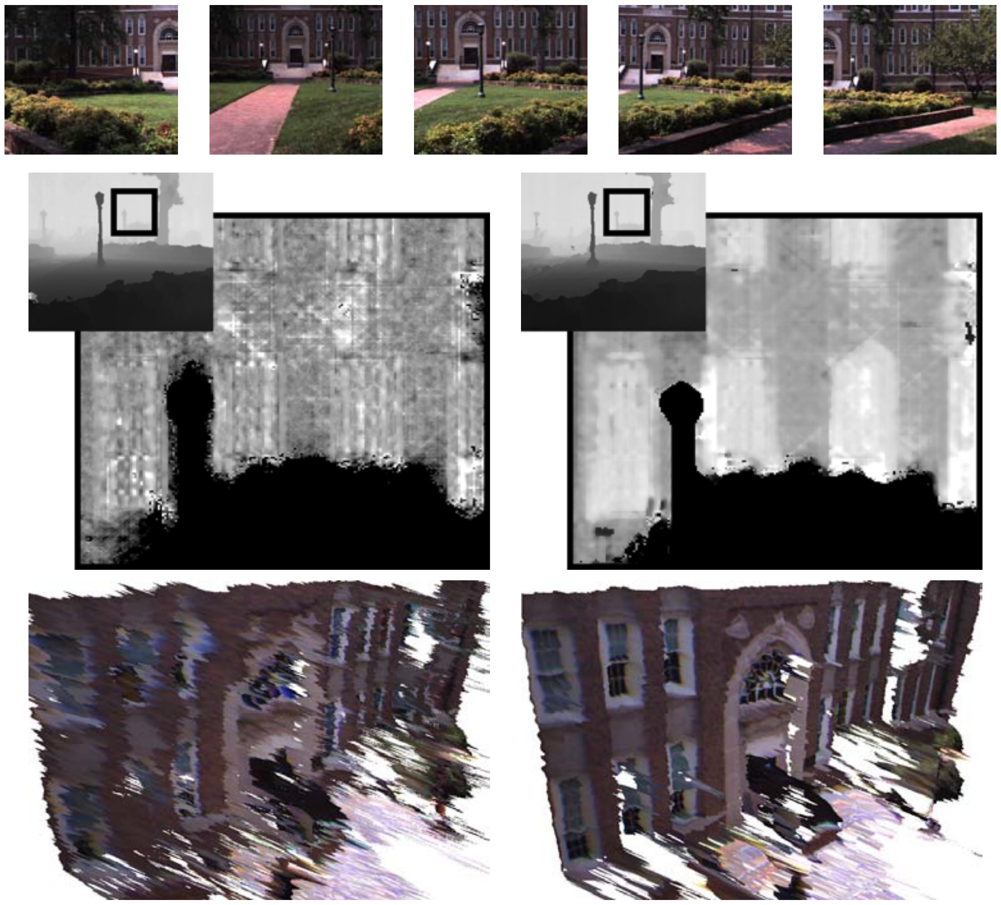
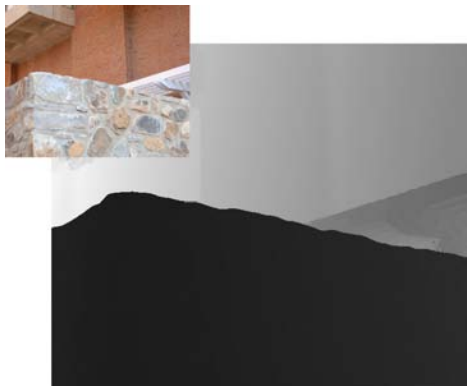
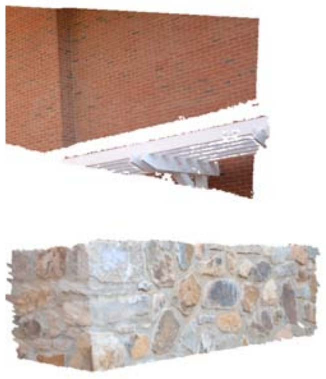


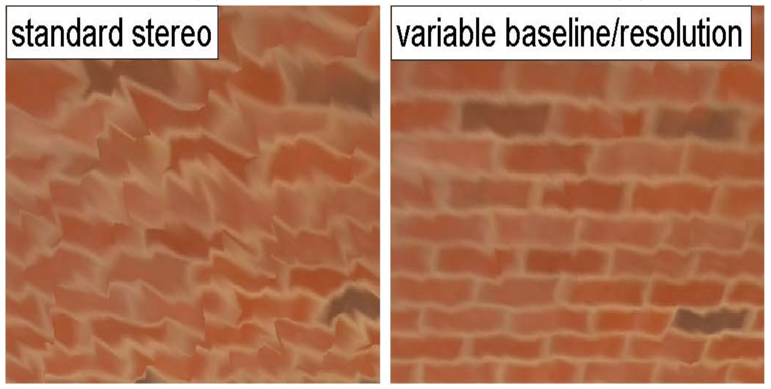
Figure 5.4: *First Row*: Some original images. The middle image is the reference view. *Second Row, Left*: Depthmap computed using standard stereo. *Second Row, Right*: Depthmap computed using our method. *Third Row*: 3D model views of standard stereo and our method. Because the correspondence accuracy is similar for both methods, the full views of the depthmaps appear similar. However, a close-up view of the standard stereo depthmap at the far range reveals the poor depth accuracy. In contrast, the close-up view of the depthmap computed using our method is much smoother, the indentations from the windows are more defined, and consequently the 3D model views are much cleaner, especially in the far range. (©2008 IEEE.)



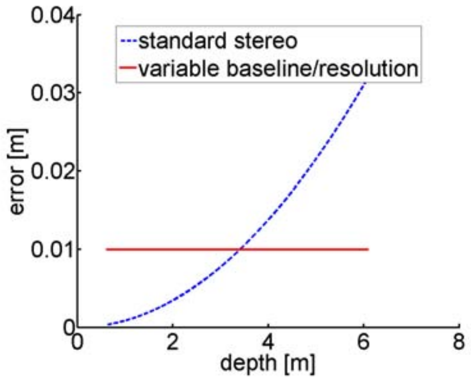
(a)



(b)



(c)



(d)

Figure 5.5: *(a)*: The reference view image and depthmap from our method. *(b)*: 3D model view from our method. *(c)*: Close-up 3D model views of the far range for standard stereo and our method. *(d)*: Depth resolution plot. (©2008 IEEE.)

	Scene 1	$z_{near} = 3m$, $z_{far} = 45m$, $\Delta_z = 30cm$			
	Fixed1	**F1/V**	Fixed2	**F2/V**	Variable
resolution	0.8 Mp	**1**	15.2 Mp	**19.0**	0.8 Mp
# pixel comps	$3.76\text{x}10^8$	**6.16**	$3.19\text{x}10^{10}$	**523**	$6.10\text{x}10^7$
resolution at z_{far}	1.32m	**4.40**	0.3m	**1**	0.3m
z where res. $= \Delta_z$	21.47m	**0.48**	45m	**1**	45m

	Scene 2	$z_{near} = 0.6m$, $z_{far} = 6.1m$, $\Delta_z = 1cm$			
	Fixed1	**F1/V**	Fixed2	**F2/V**	Variable
resolution	10 Mp	**1**	103 Mp	**10.3**	10 Mp
# pixel comps	$1.75\text{x}10^{10}$	**6.65**	$5.76\text{x}10^{11}$	**219**	$2.63\text{x}10^9$
resolution at z_{far}	0.032m	**3.20**	0.01m	**1**	0.01m
z where res. $= \Delta_z$	3.41m	**0.56**	6.1m	**1**	6.1m

Figure 5.6: This table compares our algorithm, Variable, to two versions of standard stereo, Fixed1 and Fixed2. Both Fixed1 and Fixed2 use the widest baseline possible where the near range, z_{near} is still in view. In Fixed1, the resolution is not allowed to exceed that of the actual camera, while in Fixed2, the hypothetical resolution is computed so that the error bound Δ_z is met at z_{far}. This resolution is much too high to be realized in practice. Compared to Fixed1, our algorithm is about 6 times faster and about 3-4 times more accurate at z_{far}.

$6.1m$, a maximum triangulation angle of 6°, and used 7 views for each plane. Compared to standard stereo using the widest baseline possible, our method is more than 6 times faster, and more than 3 times more accurate at the far range.

Again, we allowed standard stereo to use the widest baseline possible, so long as objects in the near range are kept in view. For the two scenes, the nearest object is 5-10% of the distance to the farthest object, which we have observed to be typical in outdoor ground-level imagery. Note that the near range has a significant effect on standard stereo, as it limits the baseline and increases disparity range. In contrast, this variable has negligible effect on our method because the baseline is variable, and because these near-range depths are processed at low resolution. For standard stereo, the resolution is insufficient to meet the target accuracy throughout the volume. In fact, the depth resolution degrades below Δ_z at about 50% of z_{far}, and grows to nearly 3-4 times worse than Δ_z at z_{far}. Our method on the other hand maintains the target accuracy throughout the volume, while still performing about 6 times faster for both scenes. Now suppose the images were captured at a resolution sufficient for standard stereo to meet the target accuracy. This resolution would be 10 to 20 times greater than that required by our method, and the processing time would be 200 to 500 times greater. Keep in mind that while the time complexity of both algorithms is proportional to ${\Delta_z}^{-3}$, that of our algorithm is proportional to ${z_{far}}^3$ as opposed to ${z_{far}}^6$, which is a dramatic improvement, especially for scenes with large depth ranges.

In our experiments, we have found an angle of triangulation of between 6 and 10 degrees to work best for our scenes. Larger angles can reduce the resolution required to meet the accuracy goal (see β in Equation 5.8), but matching is more difficult, and mismatches are more frequent.

5.6 Discussion

We have presented our Variable Baseline/Resolution Stereo algorithm, which varies the baseline and resolution proportionally with depth in order to maintain constant depth resolution throughout the reconstructed volume. This is in contrast to traditional fixed-baseline stereo in which the error increases quadratically with depth. Our approach directly addresses the accuracy and efficiency needs of an application designer wishing to employ stereo as a measuring device, and produces depthmaps which meet the desired accuracy while requiring orders of magnitude less computation than standard stereo. We have demonstrated our algorithm on real scenes in which our algorithm performs many times more accurately and efficiently than what is possible with standard stereo.

Controlling the geometric factor of depth error is is important for urban reconstruction from ground-level, because of the large depth range observed. Not only can the far range be reconstruction with greater precision, but computational effort in the near range can be reduced considerably, which is important for efficiently processing large amounts of video.

Knowing the uncertainty of the depth measurements is also important for the fusion step in Chapter 7, and having uniform depth uncertainty can simplify the algorithm and determine the resolution of the fusion volume.

At this time, the view selection algorithm in this chapter is designed only for a fronto-parallel plane-sweep. While in theory, the multi-sweep approach could also be used, it is a non-trivial extension to the current algorithm. The main reason is that for non-fronto-parallel planes, each point on the plane has a different depth, and so baseline and image resolution must be selected per pixel. This complicates the GPU implementation considerably. Selection of image resolution has been implemented with an image scale pyramid (called a mip-map in graphics processing), and until recently, mip-map levels were determined automatically and could not be selected according to depth as is required by our algorithm. Therefore combining the two approaches is currently left to future work.

One of the reasons for using semi-global matching (Hirschmuller, 2008) as opposed to real-time window-based matching, was because only fronto-parallel sweeps were used. This resulted in the slanted surfaces such as the ground to be reconstructed poorly. In the fixed baseline case, the projective distortion due to slant would fall off towards the far range as the triangulation angle decreased. But because the triangulation angle is held constant with a variable baseline, the effects of projective distortion were worse than usual. Semi-global matching was employed to remove the need for a matching window and allow pixel-to-pixel matching. Combining variable baseline/resolution stereo with multiple sweeping directions would remove this problem.

CHAPTER 6

Piecewise Planar and Non-Planar Stereo

6.1 Introduction

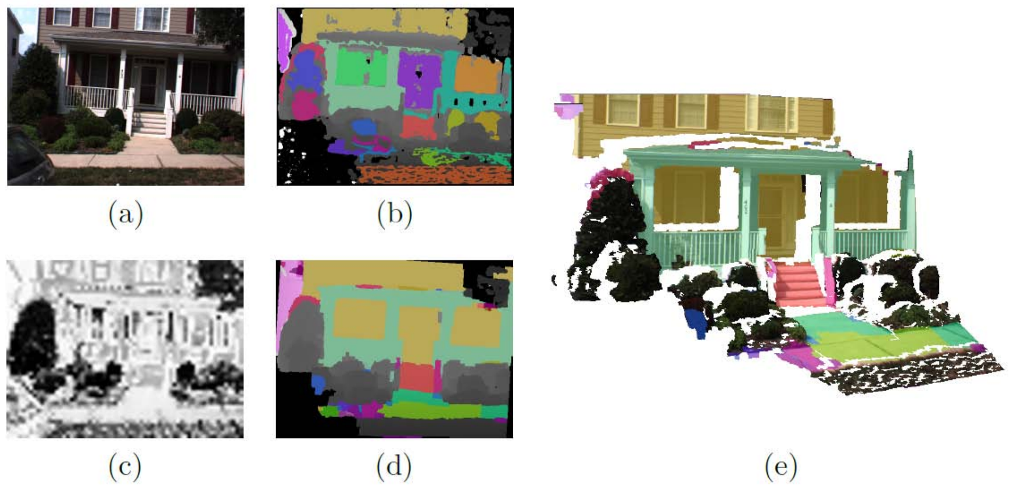


Figure 6.1: (a) Original image. (b) Planes found by RANSAC in depthmap. (c) Planar class probability. (d) Final plane labeling overlaid on depthmap. Colors = planes, gray = no plane, and black = discarded. (e) Resulting 3D model with planes highlighted. (©2010 IEEE.)

In this chapter I will present a method for segmenting an urban scene into piecewise planar and non-planar regions. Fitting a scene with a piecewise planar model has become popular for reconstructing urban scenes (Furukawa et al., 2009a; Sinha et al., 2009; Zebedin et al., 2008), as it has several advantages. The planarity assumption helps recover surfaces that are difficult to match, such as textureless or specular surfaces that often occur in urban scenes. The resulting models have a lower complexity, which is important for storing and rendering large-scale urban scenes. Furthermore, simplified geometry can often *look better*, even if the overall surface accuracy is lower, since piecewise planar surfaces better resemble man-made urban structures.

However, a piecewise planar model performs poorly in the presence of highly non-planar objects such as trees, cars, bushes, and other clutter present in urban scenes. Recent work such as Furukawa et al. (2009a) and Sinha et al. (2009) produces very convincing 3D reconstructions for man-made architectural scenes, as long as the scene is piecewise planar. But for non-planar objects and clutter, the reconstructions can appear unnatural or even completely incorrect. To address this problem, we present a stereo method capable of handling more general scenes containing both planar and non-planar regions. Our proposed technique segments an image into piecewise planar regions as well as regions which are labeled as non-planar. The non-planar regions are modeled by the output of a standard multi-view stereo algorithm. Thus, our method maintains the advantages of piecewise planar stereo, while also having the ability to fall-back to a general representation to handle non-planar surfaces.

The inputs to our algorithm are a video sequence, calibration parameters, camera poses, and depthmaps as captured by the Urbanscape system. Depthmaps can be computed using the plane-sweep method of Chapter 4. The output is a refined set of piecewise planar and non-planar depthmaps that are then used to generate a textured polygonal mesh representation of the scene. The following is an overview of our method, which will be explained in more detail in the rest of this chapter.

From the initial set of depthmaps, a number of plane hypotheses are found using a RANSAC method (Figure 6.1b). Similiar to Furukawa et al. (2009a) and Sinha et al. (2009), for each input depthmap, we set up a Markov random field (MRF) problem where each pixel is assigned a label corresponding to one of the previously obtained plane hypotheses. The key difference in our approach is the addition of a *non-plane label* which represents the input stereo depthmap. Label likelihoods are defined as the photoconsistency of the plane, in case of a plane label, or of the depthmap, in case of the non-plane label. In the spirit of model selection, the non-plane label incurs an additional penalty, due to the higher degrees of freedom in the depthmap surface. A smoothness prior is defined that penalizes label transitions and is weighted by surface continuity and image gradients. The resulting energy functional is minimized using graph-cuts (Boykov and Kolmogorov, 2004; Boykov et al., 2001; Kolmogorov and Zabih, 2002; Szeliski et al., 2006).

To further help distinguish planar and non-planar surfaces, we have trained a classifier based on image color and texture features. The training set includes image segments that have been hand-labeled as either planar or non-planar. A k-nearest neighbor classifier then produces a planar class membership probability for each segment of the oversegmented input images (Figure 6.1c), and the probability is included in the label likelihood. The reason for this additional constraint derives from our piecewise planar assumption. It may very well be that a plane fits a bush or sloping ground, at least within the precision of the stereo reconstruction. It is in fact the appearance of these image regions that indicate they are non-planar. This constraint also helps to ensure the correct plane label for specular surfaces such as windows which may have poor photoconsistency. The ability to distinguish between planar and non-planar surfaces based on appearance is due to urban scene structure. It is observed that manmade structures are often planar and exhibit regular texture. Non-manmade structures, e.g. vegetation, are typically not planar, and the texture is irregular. Color is also an important feature.

Additionally, our method links and fuses the initial plane hypotheses across overlapping views, ensuring a consistent 3D reconstruction over an arbitrary number of images. Plane segments for which the point-to-plane distances fall within a certain threshold are linked, and the plane estimates are fused. Because overlapping views can share the same plane hypotheses, the image labeling can be performed independently for each view, and the resulting 3D model will be consistent, i.e. a single planar surface can be extended indefinitely. Also, because each image is processed separately, our algorithm is *out-of-core*, needing only enough memory for one view at a time, and is therefore highly scalable. Using our system, we have reconstructed thousands of frames of street-level video. Results show our method successfully recovers piecewise planar surfaces alongside general 3D surfaces in challenging scenes containing large buildings as well as residential houses.

Compared to the methods in the other chapters in this dissertation, the method presented in this chapter is not real-time. Nevertheless, more optimization could result in near real-time performance. Memory usage is low, and few data dependencies exist between frames, so the algorithm is still scalable and could be distributed to achieve better performance.

The work in this chapter has been published in Gallup et al. (2010).

6.2 Related Work

In this section, we recall related work and discuss how it specifically applies to this chapter.

Furukawa et al. (2009a) use a very specific Manhattan-world model, where all planes must be orthogonal, and Sinha et al. (2009) use a general piecewise planar model. Non-planar surfaces are not handled well and are either reconstructed with a staircase appearance or are flattened to nearby planes. The work of Zebedin et al. (2008) focuses on aerial imagery, and in addition to planar rooftops allows for a surface of revolution representation to handle domes and spires. Our model allows for a general depthmap reconstruction as an alternative to planes, which handles any non-planar surface.

Zebedin et al. (2008) require each building to be segmented in the input, and each building is processed independently, making it trivial to scale to large datasets. Furukawa et al. (2009) present a Manhattan-world fusion technique for the purpose of generating floor plans for indoor scenes. Multiple views must be fused in a single volumetric representation, limiting the overall size of the reconstruction. We use a multi-view plane linking approach which allows images to be processed separately (out-of-core), and can produce consistent reconstructions over datasets of arbitrary size.

The use of color and texture features to classify planar surfaces is inspired by Hoiem et al. (2005) and Saxena et al. (2007) who use color, texture, and other image features to infer geometric context. They are able to create a plausible 3D representation from a single view, however no depth measurements are made. We use many of the same features as Hoiem et al. (2005), although our classification problem is much simpler. Our planar versus non-planar classifier is used in addition to photoconsistency and smoothness constraints in the image labeling task. Xiao et al. (2009) use trained classifiers and multi-view constraints to

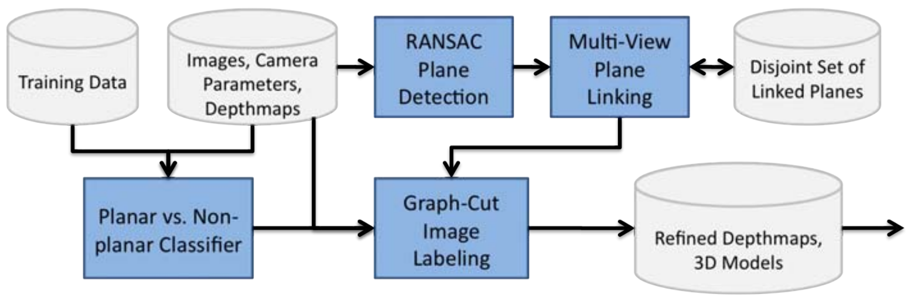

Training Data
Images, Camera Parameters, Depthmaps
RANSAC Plane Detection
Multi-View Plane Linking
Disjoint Set of Linked Planes
Planar vs. Non-planar Classifier
Graph-Cut Image Labeling
Refined Depthmaps, 3D Models

Figure 6.2: Our piecewise planar and non-planar stereo system. (©2010 IEEE.)

segment street-side images into ground, tree, building, and sky regions. 3D models are then fit to the buildings. We also combine learned image appearance with multi-view constraints, but make no hard decision until the the final plane labeling.

6.3 Piecewise Planar and Non-Planar Stereo

The steps of our algorithm are laid out in Figure 6.2. The input to our algorithm is a collection of images, camera poses, and depthmaps. Depthmaps are computed using the real-time plane-sweeping in Chapter 4. This reconstruction is typically poor for weakly textured and specular surfaces, but the result is sufficient to initialize our algorithm.

6.3.1 Plane Hypothesis Generation

We first obtain a set of plane hypotheses for every image using a RANSAC method. Typically one seeks to find a single model to fit all the data, but our objective is to find multiple locally fit models. In this regard, there are several important aspects of our method that are crucial to achieving a good set of planes.

- **Sampling.** A plane model can be obtained from three points in the depthmap sampled at random. The first point is selected from a uniform distribution over the image. The other two points are selected from normal distributions centered at the first point with a standard deviation of σ.

- **Scoring.** Each model is only evaluated against points nearby the original samples. Specifically, only points within M pixels of the first sample are considered. Instead of scoring simply by the inlier count (number of points within a threshold distance to the plane), we score by the likelihood of each point fitting the plane, according to the MLE-sac method (Torr and Zisserman, 2000).

- **Contiguity.** After RANSAC returns a plane, the inlier set is determined by computing the distance of each point to the plane. Additionally, the inlier set is restricted to points that are connected (contiguous) to the initial sample, according to the image graph. A new plane is obtained as the least-squares fit to the inlier points. Inliers are again determined, and the process is repeated for several iterations. (This contiguity constraint is not used inside the RANSAC sampling loop for performance reasons.)

The final set of inliers is then removed from the image, and RANSAC is again repeated on the remaining points. For each image we obtain a set of N planes $\Pi = \{\pi_1 \cdots \pi_N\}$. This tends to include most of the major planes in the scene, as well as some spurious planes which happen to fit well to non-planar or quasi-planar structures. We add to each set the plane at infinity, denoted π_∞, which is useful for labeling sky or distant surfaces which are not reconstructed by stereo. At this point, an initial labeling of each image can be performed, simply by assigning each inlier set from RANSAC to its respective plane.

For all of our experiments we use $\sigma = 8$ pixels, $M = 100$ pixels, and $N = 20$ planes. Note that our plane detection method is much simpler than that of Sinha et al. (2009). One reason is that that method operates on points and lines while our method operates on depthmaps.

6.3.2 Multi-View Plane Linking

For multi-view reconstructions, it is imperative to obtain consistent plane hypotheses across overlapping views. A planar surface visible in several images will generate slightly varying plane hypotheses, due to small variations in the depthmaps. Also, it is intractable to consider every plane for every image when processing large datasets. Thus we perform a single pass over all images and establish links across nearby views between mutually supporting planes. All linked plane hypotheses are fused to give a single multi-view estimate for the plane.

Planes are linked as follows. For every plane π_i, the set of all planes in nearby views, including the planes in the same view as π_i, is considered for linking. For every plane π_j in that set, if a sufficient number of points (90%) belonging to π_i falls within a threshold distance (1% of the camera-to-point distance) of π_j, then π_i and π_j are linked. A new plane is fit to the combined set of points belonging to all the linked planes. A global disjoint set data structure is created which maintains each set of linked planes. The disjoint set can be held in memory at all times, since only a few bytes are required to identify a plane. This ensures that surfaces seen in multiple images have the exact same plane hypothesis. It also serves to link similar planes from repeated structures, or single planes which appear disjoint in the images due to occlusion. See Figure 6.3.

6.3.3 Graph-Cut Labeling

Once the plane hypotheses have been established, the next step is to perform a pixel-wise labeling of each image. Each image is processed independently, but since plane hypotheses have been fused, the resulting depthmaps will be globally consistent. For each image, an MRF is defined, leading to a standard energy minimization problem involving data and smoothness terms. Our goal is to obtain a labeling so as to minimize the energy functional

$$E(L) = \sum_{p \in \mathcal{I}} E_{data}(L(p)) + \sum_{p,q \in \mathcal{N}} \lambda_{smooth} E_{smooth}(L(p), L(q)), \tag{6.1}$$

where $\mathcal{I}$ is the set of pixels in the image, $\mathcal{N}$ is the standard 4-neighborhood, and L is the labeling.

The set of labels is the union of all plane labels, a *non-plane* label, and a *discard* label. The labeling function $L : \mathcal{R}^2 \to \{\pi_1, \cdots, \pi_N, \pi_\infty, \textit{non-plane}, \textit{discard}\}$ identifies a label for every pixel. The *non-plane* label indicates the original stereo depthmap, and the *discard* label indicates no reliable reconstruction could be obtained.

The E_{data} function is defined as

$$E_{data}(l) = \begin{cases} \min(\rho(l), \rho_{max}) & \text{if } \ l \in \{\pi_1 \cdots \pi_\infty\} \\ \min(\rho(l), \rho_{max}) + \rho_{bias} & \text{if } \ l = \textit{non-plane} \\ \alpha\rho_{max} & \text{if } \ l = \textit{discard} \end{cases} \tag{6.2}$$

where ρ is a photoconsistency (dissimilarity) measure between the pixels in nearby views put into correspondence by the assigned plane (using a homography), or by the original stereo depthmap. For photoconsistency we use the Birchfield-Tomasi pixel-to-pixel dissimilarity measure (1998). For occlusion handling we use the multi-view technique of Kang et al. (2001). For the *non-plane* label, a penalty ρ_{bias} is given in order to penalize models with more degrees of freedom. The dissimilarity measures have been truncated to ρ_{max} in order to handle poorly matching specular or reflective surfaces such as windows. The *discard* label receives slightly less penalty than the maximum. Thus small, poorly matching regions will be labeled according to their surroundings due to the smoothness term, but large poorly matching regions will incur enough cost to be discarded. For all our experiments we set $\rho_{max} = 6$, $\rho_{bias} = 0.5$, and $\alpha = 0.9$. (See Figure 6.3.)

The E_{smooth} function is defined as

$$E_{smooth}(l_p, l_q) = g \cdot \begin{cases} 0 & \text{if } \ l_p = l_q \\ d_{max} & \text{if } \ l_p \text{ or } l_q \in \{\pi_\infty, \textit{discard}\} \\ d' & \text{otherwise} \end{cases} \tag{6.3}$$

$$d' = \min(d, d_{max}) + d_{min} \tag{6.4}$$

$$g = \frac{1}{\gamma \|\partial I / \partial u\|^2 + 1} \tag{6.5}$$

where d is the distance between the 3D neighboring points according to their labels, and g is the image gradient magnitude (color or grayscale) between the two neighbors. Our video sequences were captured along with GPS data, so absolute distances can be measured. Otherwise, distances can be defined relative to the median value in the depthmap for example.

The term d_{min} is a minimum penalty given in order to prevent spurious transitions between planes that are close to each other in 3D. The term d_{max} makes the penalty robust to discontinuities. For all our experiments we set $\lambda_{smooth} = 5$, $d_{min} = 2$, $d_{max} = 0.2$ meters, and $\gamma = 10$.

The energy can be minimized using the well-known multi-label graph-cut method (Boykov et al., 2001). One limitation of graph-cuts, and discrete MRFs in general, is that of metrication, which follows a Manhattan distance, not a Euclidean one. This leads to stair-case and other artifacts. However, we use this to our advantage in man-made scenes, where the vertical direction and dominant facade normal can be readily obtained as in Section 4.4.1. The image can then be rectified so that the horizontal and vertical vanishing points correspond to the x and y axes. Then the Manhattan distance metrication actually helps to enforce that label boundaries follow vertical and horizontal lines.

6.3.4 Planar Classifier

Even with the *non-plane* label available, surfaces such as bushes, trees, and grass are occasionally detected and assigned to planes by the graph-cut solution. Ultimately for some regions, within the uncertainty of the stereo depth, a plane may well fit those surfaces. To handle such cases, we train a classifier based on color and texture features to distinguish between surfaces that are typically planar, and those that are not. Features are computed from image patches. Inspired by Hoiem et al. (2005), we use the following color features: mean red, green, and blue (RGB color space), mean hue, saturation, value (HSV color space), and the hue histogram (5 bins). We use the following features computed from the edge orientation histogram (Kumar and Hebert, 2003): entropy, maximum value, and number of modes. The texture features capture the fact that man-made objects tend to have only a few consistent edge orientations, while natural objects have a less structured appearance.

Each image is segmented into a grid of 16×16 pixel cells, and the feature vector is computed for each cell. We experimented with commonly used oversegmentation (superpixel) algorithms (Ren and Malik, 2003; Felzenszwalb and Huttenlocher, 2004b), but in the end we preferred the regular grid. We have found that there is enough information in the photoconsistency and smoothness penalties to find accurate object boundaries. Therefore we prefer the grid as it ensures segments of a uniform size and density. The training data consists of approximately 5000 segments in 5 images which were labeled by hand as planar or non-planar. For a given input image, the planar class probability for each grid cell is computed using k-nearest-neighbors. In this step we are interested in the class membership probability, computed as the percentage of features labeled *planar* in the neighbor set, and we will defer the final plane labeling decision until the graph-cut.

Let $a \in [0, 1]$ be the planar class probability for a given segment, and l be the label of a pixel within that segment. The data term now becomes

$$E'_{data}(l) = E_{data}(l) + \lambda_{class} \begin{cases} 1 - a & \text{if } \ l \in \{\pi_1, \cdots, \pi_\infty\} \\ a & \text{if } \ l = \textit{non-plane} \\ 0 & \text{if } \ l = \textit{discard}. \end{cases} \tag{6.6}$$

We have set $\lambda_{class} = 2$ for all our experiments. Figure 6.4 demonstrates the effect the class penalty has on the labeling result.

6.4 Experiments

To test our system, we have processed street-side video captured by the Urbanscape system. The video was captured by two vehicle-mounted Point Grey Flea2 1024×768 color cameras. The cameras are aimed perpendicular to the driving direction, with one camera pointed horizontally and the other pointed upwards at 30 degrees. The composite camera system has a horizontal field of view of 120 degrees, and a vertical field of view of 60 degrees. The captured data contains a variety of street-level scenes with large buildings and residential houses. These scenes contain large planar facade surfaces but also many non-planar objects such as bushes, trees, and cars.

For all our experiments we have used the parameter values that have been given throughout Section 6.3. Parameters were chosen empirically and without much difficulty. Also, the same set of training data was used for all experiments. The fact that we used the same set of parameters and training data for several diverse datasets indicates that they are not overly critical.

Although the end goal of our approach is to produce depthmaps, we have evaluated the final graph-cut labels for accuracy against a hand-labeled test set of 31147 planar and non-planar image segments in 28 images. Since the graph-cut labels are computed on the full resolution, the segment label is determined by majority vote. Any of $\{\pi_1, \cdots, \pi_N, \pi_\infty\}$ count as planar, *non-plane* counts as non-planar, and *discard* is not counted. 96.1% of the planar segments and 86.0% of the non-planar segments were labeled correctly–an average accuracy of 91.1%. (We will focus on the average because there are many more planar test examples than non-planar ones.) We have performed additional evaluations using the classifier only, using the graph-cut labeling without the classifier, and using the classifier with only texture or color features. The results of these experiments are shown in Figure 6.5. Of particular interest is the fact that the graph-cut labeling achieves an average classification rate of 64.4% without the classifier, 78.6% with the classifier trained with texture features, 89.2% with the classifier trained with color features, and 91.1% with the classifier trained with both types of features. This shows that the classifier has a significant effect, and while color is clearly the most important type of feature, the texture features are also quite effective, and they improve the accuracy. It is also interesting that planar classification rate using the graph-cut labeling with the classifier is *lower* than when using the classifier alone. This appears to be because not all planar surfaces are discovered by RANSAC, and the graph-cut labeling must have a correct plane hypothesis in order to produce a planar label.

Figure 6.6 shows 9 images sampled from our results. For each image, the results of the RANSAC plane detection, planar classification, and graph-cut labeling are shown. In each scene, most of the major planes are found by our RANSAC method, although some planes are occasionally missed, especially when they occupy only a small portion of the image. The planar classifier performs well, despite its simplicity, and provides a good cue for the final graph-cut labeling to select between plane labels and the *non-plane* label. The final graph-cut labeling recovers broad planar surfaces while also identifying non-planar surfaces. Note that even though the planar classifier is run on a coarse grid, the graph-cut result recovers fine object boundaries due to the photoconsistency constraint.

The number of input video frames ranges from 200 to 800 (see Figures 6.7-6.9 for exact numbers), and a refined depthmap is computed for every 10th frame. Our unoptimized C++ implementation ran on an Intel Xeon 2.67 GHz PC and took about 1 to 2 minutes per depthmap for all steps. This has the potential to be sped up considerably by using a real-time variant for the RANSAC step such as ARRSAC (Raguram et al., 2008) and a faster MRF solver than graph-cuts such as the variational GPU-based meth of Zach et al. (2008)

Figures 6.7-6.9 show the final 3D models produced by our system. Many textureless and specular surfaces that were missed in the original reconstruction were recovered by our system due to the piecewise planar model. Also, because our model enforces planes, straight lines on planar surfaces remain straight in the 3D models. Especially note that the non-planar surfaces are preserved, and are not flattened to planes as in other piecewise planar stereo methods.

6.5 Discussion

In this chapter I have presented a piecewise planar reconstruction method that also handles non-planar surfaces. Planar and non-planar surfaces are segmented with the aid of a classifier that exploits the structured texture and color of manmade objects in urban scenes. Results have shown that our piecewise planar and non-planar method can successfully recover planar surfaces alongside non-planar surfaces, even in highly cluttered scenes. One of the weaknesses of our reconstructions is the lack of completeness. Many planar surfaces that occupy only a small part of the image are missed by our system, and other surfaces are simply not seen in any of the cameras. This can be addressed by adding a more complete set of views to the dataset. Note that the scenes considered in our work are significantly more cluttered than those addressed by previous piecewise planar stereo methods.

One direction for future work would be to use more surface primitives and classes. Besides planes, other simple shapes appear in urban scenes like cylinders, spheres, and cones. The RANSAC plane detection step could be extended to search for these objects as was done in Labatut et al. (2009). Also, using a richer set of classes besides planar/non-planar could also help the reconstruction. For example, explicitly detecting glass windows would be beneficial since stereo performs poorly on these reflective surfaces. This quickly leads to the problem of image parsing where labeling the image with its semantic or geometric class is the primary goal. Combining our stereo system with a more sophisticated parsing method, such as Tighe and Lazebnik (2010), could possibly improve reconstruction and labeling simultaneously.

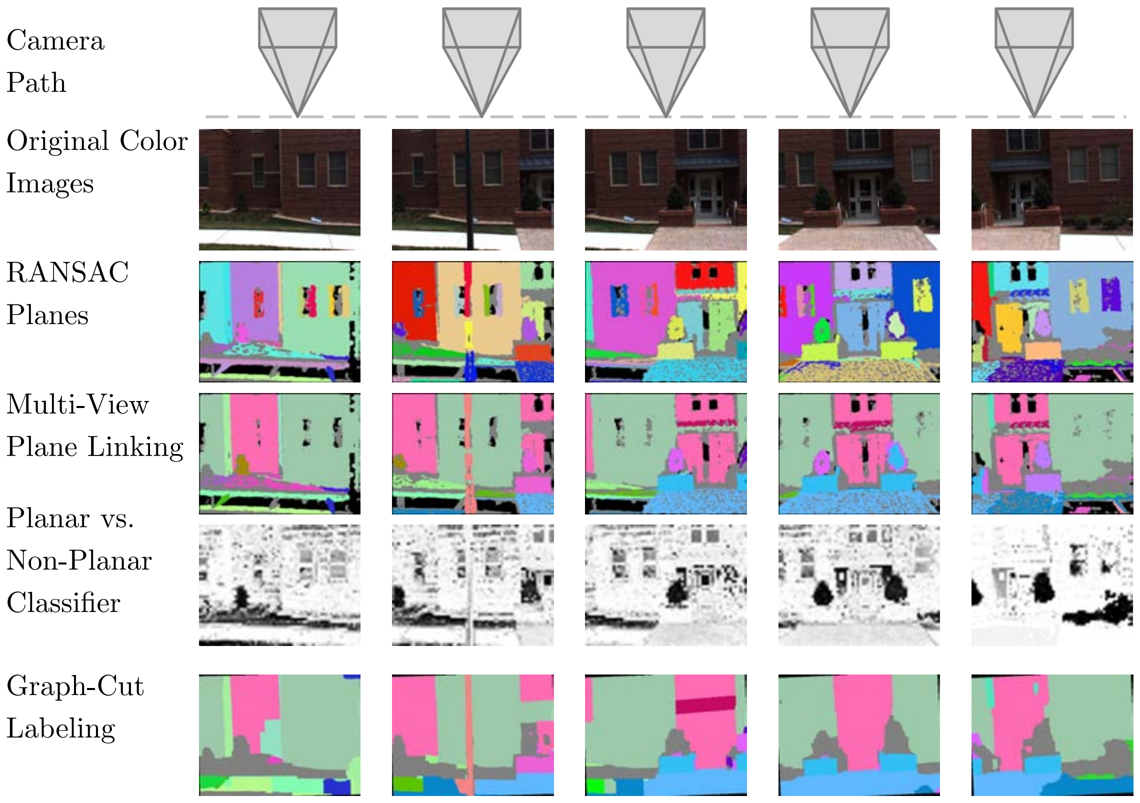


Figure 6.3: The second row shows a subset of the images used to compute the depthmaps. The third row illustrates the the plane detection results of our modified RANSAC technique. The fourth row shows the plane labels after linking. Note that the same scene plane now has a consistent label. The fifth row shows the classification results of the image into planar or non-planar structure (planar class probability black=0, white=1). The sixth row shows the results of the graph cut based plane assignment. See Figure 6.7 for the resulting 3D model. (©2010 IEEE.)

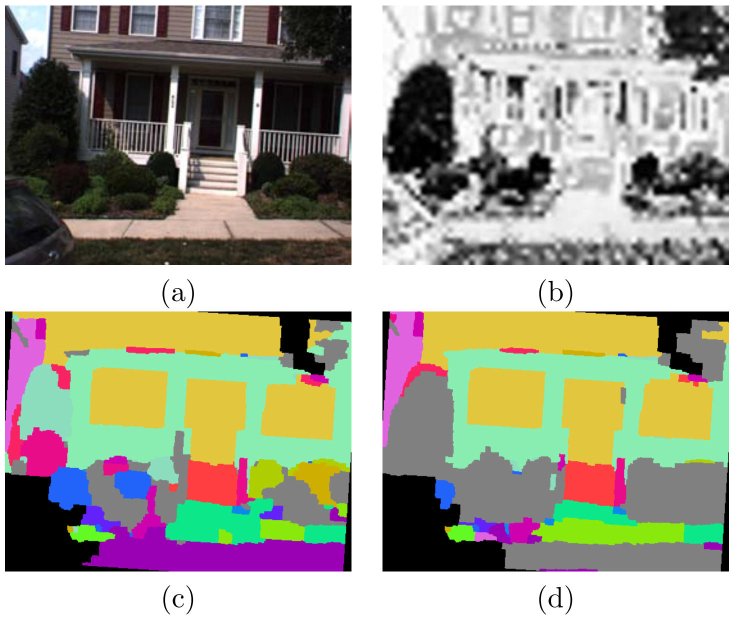


Figure 6.4: Result of graph-cut labeling with and without the planar class probability term. (a) Original color image. (b) Planar class probability. (c) Graph-cut labeling result without the class probability term. (d) Labeling result with the class probability term, which helps to remove many false planes labeled in the bushes and grass. (©2010 IEEE.)

Classifier Only			
	Texture	Color	All
Planar	87.0	97.5	98.0
Non-Planar	61.8	80.6	81.3
Average	74.4	89.1	89.7

Graph-cut Labeling				
	No Classifier	Texture	Color	All
Planar	66.2	88.1	96.8	96.1
Non-Planar	62.7	69.0	81.5	86.0
Average	64.4	78.6	89.2	91.1

Figure 6.5: Classification rates of the classifier alone and the graph-cut labeling with and without the classifier. Also compared are the rates of the classifier trained with only texture features, with only color features, and with all features.

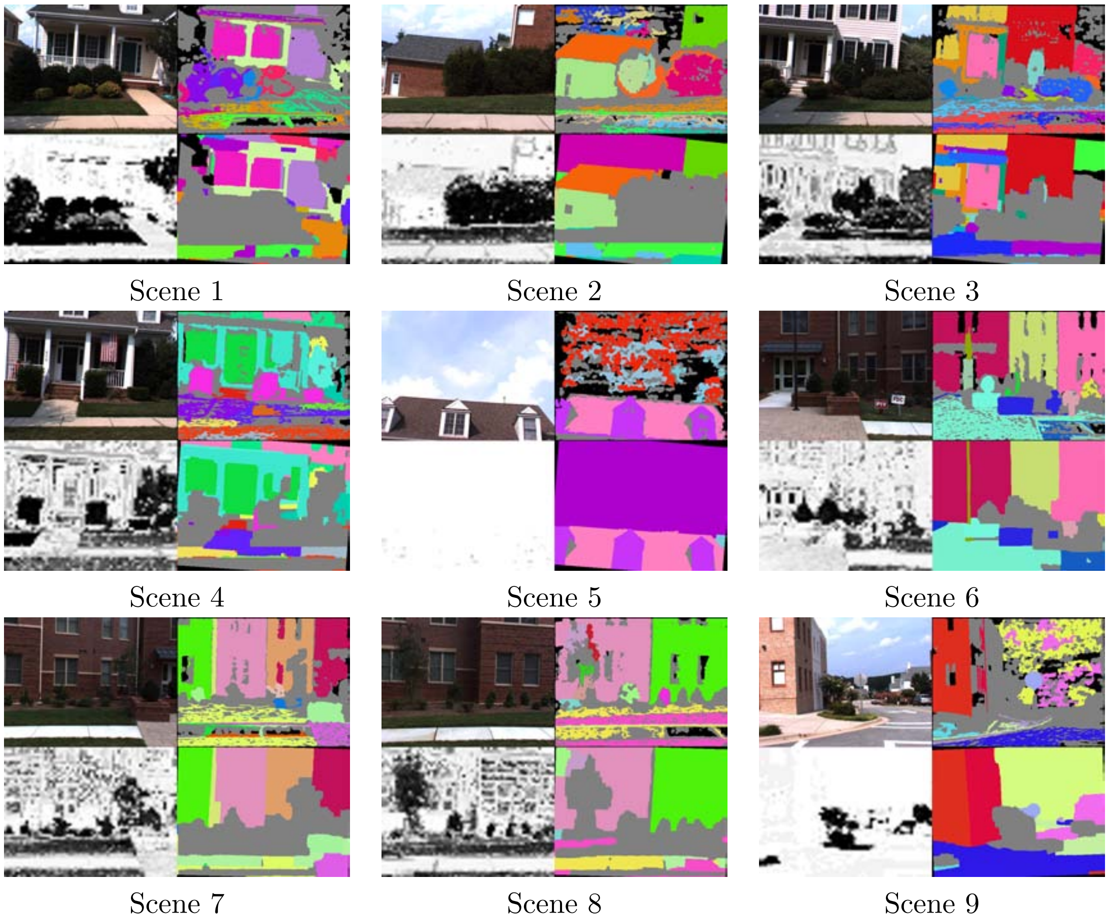
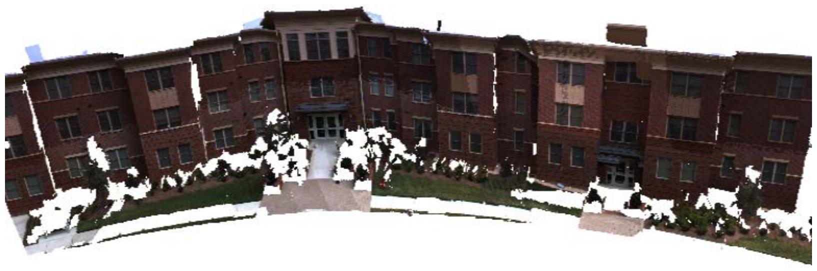
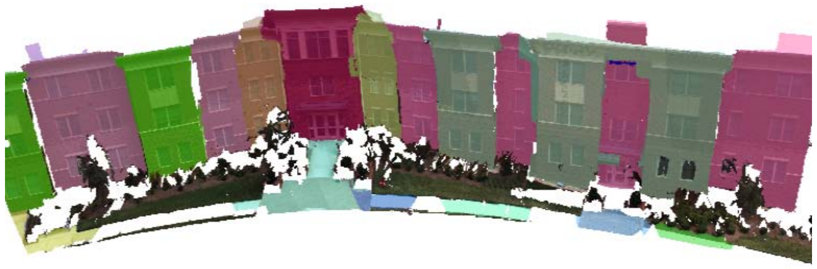


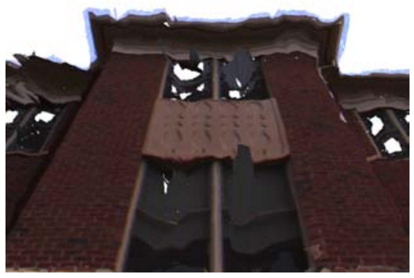
Figure 6.6: Results from the various steps of our algorithm for several scenes. Each of the four panes is as follows. Top Left: original color image. Top Right: RANSAC planes. Bottom Left: Planar class probability. Bottom Right: Final graph-cut labeling.



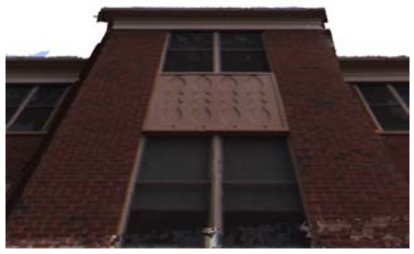
Final 3D Model



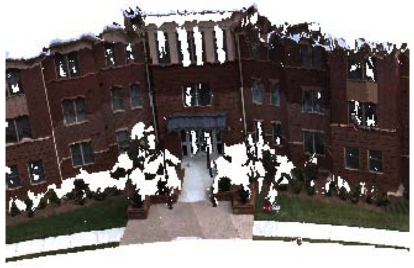
3D Model with Highlighted Planes

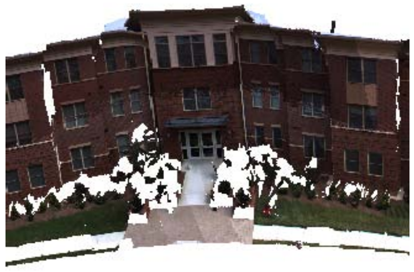


Before



After





Before

After

Figure 6.7: 3D model produced by our piecewise planar and non-planar stereo algorithm from 400 images (40 depthmaps). The before and after images show the improvements of a piecewise planar model: textureless and specular surfaces (windows) are recovered, straight lines remain straight, and 3D model complexity is reduced. Also note that the reconstruction is able to preserve non-planar surfaces as well. (©2010 IEEE.)

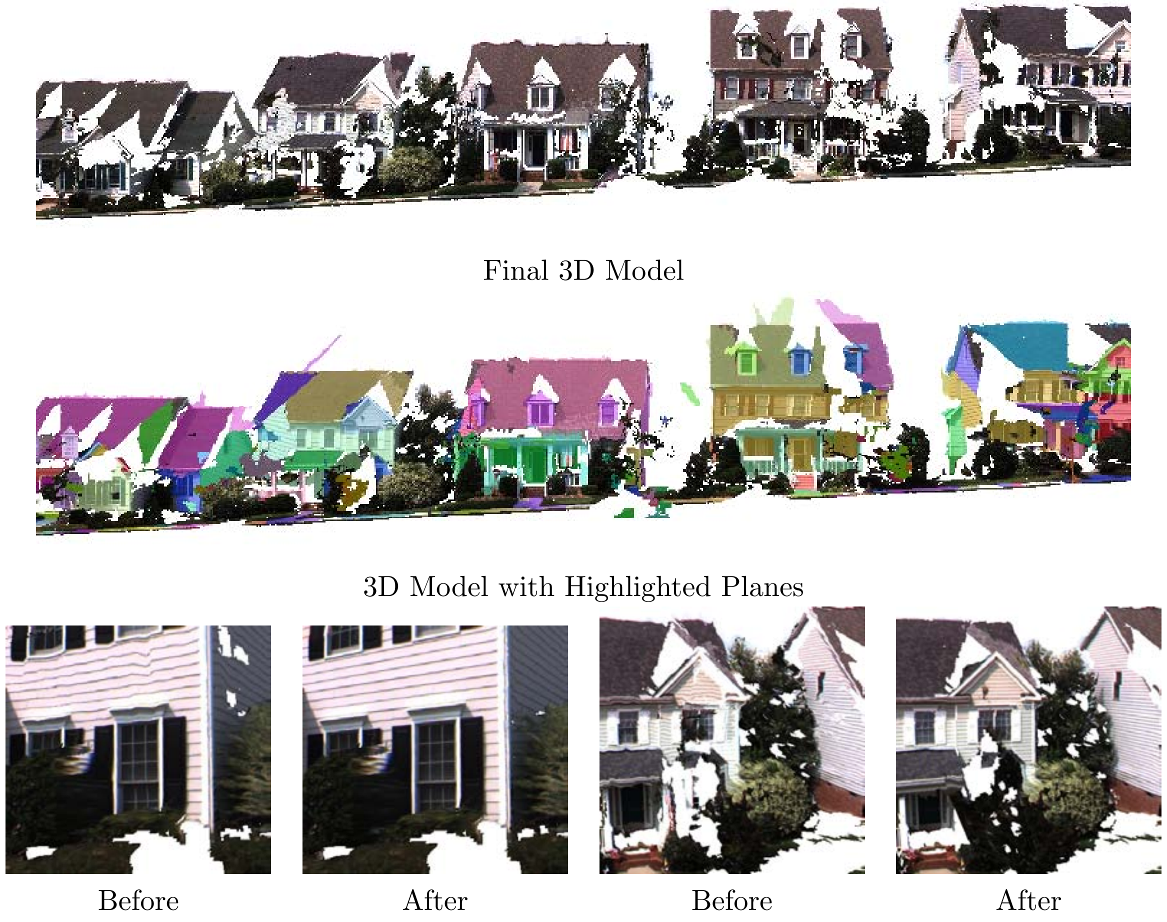


Figure 6.8: 3D model produced by our piecewise planar and non-planar stereo algorithm from 800 images (80 depthmaps). (©2010 IEEE.)

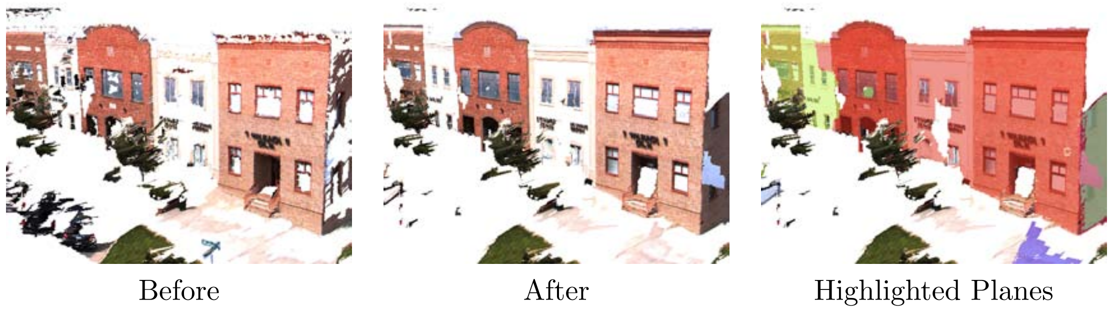


Figure 6.9: 3D model produced by our piecewise planar and non-planar stereo algorithm from 200 images (20 depthmaps). (©2010 IEEE.)

CHAPTER 7

A Heightmap Model for 3D Reconstruction

7.1 Introduction

In this chapter, I will present a volumetric surface reconstruction method that uses a heightmap representation. This addresses the problem of surface generation from depth measurements. This surface can be represented implicitly by a volumetric function or explicitly by a polygonal mesh. Surface generation was also addressed to a degree in Chapter 6, where piecewise planar segmentation formed surfaces from single views. But in this chapter, all the depth information (within a volume of interest) will be fused to form a single unified surface. The method models the occupancy probability of each point in the volume based on the depth measurements, and then extracts the surface as the boundary between full and empty space. However, rather than reconstruct a general surface, greater speed and lower memory usage can be achieved by using a heightmap constraint. In the simplest case, each vertical column is assigned a single height value that optimally separates full and empty space in that column. To handle more complex scenes, a multi-layer heightmap is used, where each layer represents a transition between full and empty space. The heights of all the layers in a column can be computed optimally with a dynamic programming method.

Compared to other volumetric methods (Zach et al., 2007), the heightmap model has several advantages. First, it enforces that walls and facades are strictly vertical, since they appear as discontinuities in the heightmap. Using a heightmap for ground-based measurements has the advantage that the estimated parameter, height, is approximately perpendicular to the dominant direction of measurement noise. This is ideal for urban

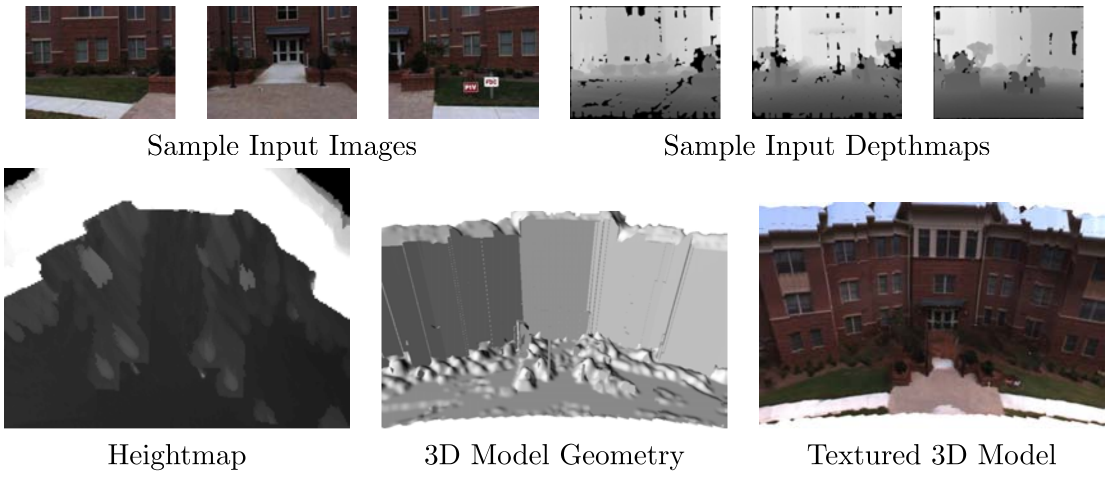


Figure 7.1: A heightmap model for 3D reconstruction.

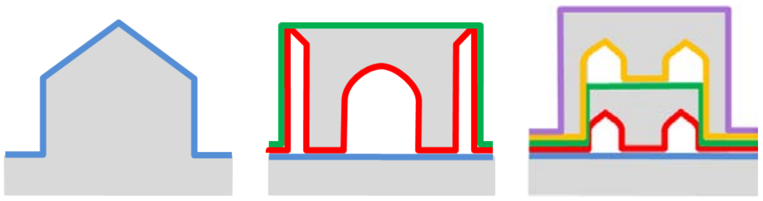


Figure 7.2: Example cross-sections of n-layer heightmaps.

reconstruction where vertical walls are of particular interest. Second, because the height estimate is constrained by all the data in the entire vertical column, good results can be obtained wihtout regularization, leading to a highly parallel and efficient computation. In fact, the occupancy probabilities of the column only need to be stored temporarily, thus avoiding the need to store the entire volume during optimization. Third, heightmaps can be stored more efficiently than polygon meshes. Therefore the heightmap representation yields efficient computation and storage.

One limitation of our model compared to a general geometric representation is that it does not accommodate complex surfaces that would require a large number of layers. Using too many layers would approach general volumetric methods and without regularization would lead to ambiguous and noisy layer positions. Thus our algorithm presents a trade-off compared to more general 3D reconstruction methods. It is more efficient, robust, and produces more compact models at the expense of losing some detail.

Our method runs on the GPU, and the complete system can process video at 13 Hz. We have demonstrated our approach on several challenging street-side video sequences from the Urbanscape project.

Another data source for urban 3D reconstruction is images downloaded from photo sharing websites such as Flickr. In this case, data acquisition is free but the datasets are usually limited to popular tourist locations. Camera poses can be computed using techniques such as the Photo Tourism system of Snavely et al. (2006) and the more recent methods of Frahm et al. (2010) and Agarwal et al. (2009). Dense stereo and surface modeling were first achieved by Goesele et al. (2007) and recently by Furukawa et al. (2010). We apply our extended heightmap approach to 3D reconstruction from community photo collections as well. Our approach is much simpler and faster, and yet results are surprisingly good.

In the rest of this Chapter, I will first present the heightmap reconstruction method in Section 7.2, then I will describe how this method can be used to perform reconstruction from video sequences and photo collections in Section 7.3.

The method presented in this chapter was published in Gallup et al. (2010a) and Gallup et al. (2010b).

7.2 n-Layer Heightmap Method

A single layer heightmap defines a surface, which is the transition from occupied space to empty space. In an n-layer heightmap, each layer defines a transition from full to empty or vice versa. Figure 7.2 illustrates single- and multi-layer heightmaps where each layer is represented by a different color. The number of layers needed to reconstruct a scene can be determined with a vertical line test. For any vertical line, the number of surfaces that the line intersects is the number of layers in the scene. In our approach, the user must give the maximum number of layers beforehand, although model selection may determine that fewer layers are sufficient.

The input to our method is a set of images with corresponding camera poses and their depthmaps. In this section we will assume that the volume of interest has been defined. Section 7.3 discusses how to lay out the volume for reconstructing video sequences and photo collections. The depth measurements from each camera are used to determine the occupancy likelihood of each point in the volume, and an n-layer heightmap is fit.

In our approach we use the probability occupancy grid from the robotics literature (Margaritis and Thrun, 1998; Pathak et al., 2007). The occupancy of each voxel is computed by Bayesian inference, and our derivation is similar to that of Guan et al. (2008). We model the measurement distribution as a combination of normal and uniform distributions in order to better handle outliers. Robustness to outliers is critical since our input measurements are stereo depthmaps.

We will now describe how to compute the probabilistic occupancy grid over the volume of interest. Since the heightmap layers will be computed independently for each vertical column of the volume, the occupancy grid does not need to be fully stored. Each column must be stored only temporarily, which keeps the memory requirement low. We will first derive the occupancy likelihood for each voxel independently. Voxel occupancy is in fact not independent since it must obey the layer constraint, and we will later show how to compute the layers for a column of voxels using dynamic programming. The variables used in our derivation are summarized as follows:

- O_p: a binary random variable representing the occupancy of voxel p.
- $Z_p = Z_1 \dots Z_k$: depth measurements along rays intersecting p from cameras $1 \dots k$.
- z_{min}, z_{max}: depth range of the scene.
- σ: depth measurement uncertainty (standard deviation).
- S: depth of surface hypothesis.
- $L_x = l_1 \dots l_n$: configuration of layers at point x in the heightmap. l_i is the vertical position of layer i.

For simplicity we have assumed that all depth measurements have the same uncertainty σ although this is not a requirement.

We will now derive the likelihood for O_p. (We will drop the subscript p until multiple voxels are considered for dynamic programming):

$$P(O|Z) \propto P(Z|O)P(O) \tag{7.1}$$

$$P(Z|O) = \prod_{i=1\dots k} P(Z_i|O) \tag{7.2}$$

Equation 7.2 states our assumption that the measurements are independent. We use the occupancy prior $P(O)$ to slightly bias the volume to be empty above the camera center and full below. This helps to prevent rooftops extending into empty space since the cameras do not observe them from the ground.

To determine $P(Z_i|O)$ we will follow Guan et al. (2008) and introduce a helper variable S which is a candidate surface along the measurement ray. The depth measurement can then be formulated with respect to S.

$$P(Z_i|O) = \int_{z_{min}}^{z_{max}} P(Z_i|S,O)P(S|O)dS \tag{7.3}$$

$$P(Z_i|S,O) = P(Z_i|S) = \begin{cases} \mathcal{N}(S,\sigma)|_{Z_i} & \text{if inlier} \\ \mathcal{U}(z_{min}, z_{max})|_{Z_i} & \text{if outlier} \end{cases} \tag{7.4}$$

$$= \rho\mathcal{N}(S,\sigma)|_{Z_i} + (1-\rho)\mathcal{U}(z_{min}, z_{max})|_{Z_i} \tag{7.5}$$

The measurement model is a mixture of a normal distribution $\mathcal{N}$ and uniform distribution $\mathcal{U}$ to handle outliers. $\mathcal{N}|_Z$ is the distribution's density function evaluated at Z. ρ is the inlier ratio, which is a given parameter. $P(S|O)$ is the surface formation model defined as follows where $\epsilon \to 0$ and z_p is the depth of the voxel.

$$P(S|O) = \begin{cases} 1/(z_{max} - z_{min}) & \text{if } S < z_p - \epsilon \\ \left(1 - z_p/(z_{max} - z_{min})\right)/\epsilon & \text{if } z_p - \epsilon \le S \le z_p \\ 0 & \text{if } S > z_p \end{cases} \tag{7.6}$$

This model states that the surface must be in front of the occupied voxel, but not behind it. We will also need the measurement likelihood given that the voxel is empty, which we will denote by $\neg O$. The derivation is the same, replacing O with $\neg O$, except that the surface formation model is

$$P(S|\neg O) = 1/(z_{max} - z_{min}). \tag{7.7}$$

We will now define our n-layer model and show how to recover it with dynamic programming. We will derive the likelihood of L_x which is the layer configuration at pixel x in the heightmap. This pixel contains a vertical column of voxels, which we will denote as O_i where i is the height of the voxel ranging from 0 to m.

$$\begin{aligned} P(L|Z) &\propto P(Z|L)P(L) && (7.8)\\ P(Z|L) &= \prod_{i=0}^{l_1-1} P(Z|O_i) \prod_{i=l_1}^{l_2-1} P(Z|\neg O_i) \ldots \prod_{i=l_n}^{m} P(Z|\neg O_i). && (7.9)\\ P(L) &= \prod_{i=0}^{l_1-1} P(O_i) \prod_{i=l_1}^{l_2-1} P(\neg O_i) \ldots \prod_{i=l_n}^{m} P(\neg O_i). && (7.10) \end{aligned}$$

Note that the measurement likelihoods alternate between the full condition $P(Z|O_i)$ and the empty condition $P(Z|\neg O_i)$ as dictated by the layer constraint. Also note that the number of layers is assumed to be odd, giving the final product the empty condition. This is true for outdoor urban scenes where the the bottom of the scene terminates with the full condition (the ground) and the top of the scene terminates with the empty condition (the sky). For indoor scenes, an even number of layers could be used.

We will now define our cost function C by taking the negative log-likelihood of $P(L|Z)$, which will simplify the dynamic programming solution.

$$\begin{aligned} C = -\ln P(Z|L)P(L) &= -\sum_{i=0}^{l_1-1} (\ln P(Z|O_i) + \ln P(O_i)) \\ &\quad - \sum_{i=l_1}^{l_2-1} (\ln P(Z|\neg O_i) + \ln P(\neg O_i)) \ldots && (7.11) \end{aligned}$$

To simplify the sums over the layers we will define the following:

$$
\begin{aligned}
I_a^b &= -\sum_{i=a}^{b} (\ln\, P(Z|O_i) + \ln\, P(O_i)) & (7.12)\\
\bar{I}_a^b &= -\sum_{i=a}^{b} (\ln\, P(Z|\neg O_i) + \ln\, P(\neg O_i))\,. & (7.13)
\end{aligned}
$$

The sums I_0^b (resp. $\bar{I}$) for all b can be precomputed making it easy to compute $I_a^b = I_0^b - I_0^{a-1}$ (resp. $\bar{I}$).

We can now write our cost function recursively in terms of C_k which is the cost only up to layer k.

$$
\begin{aligned}
C_k(l) &= \begin{cases} I_{l'}^{l} + C_{k-1}(l') & \text{if } odd(k) \\ \bar{I}_{l'}^{l} + C_{k-1}(l') & \text{if } even(k) \end{cases} & (7.14)\\
l' &= \operatorname*{argmin}_{l' \leq l} C_{k-1}(l') & (7.15)\\
C_0(l) &= 0 & (7.16)
\end{aligned}
$$

The original cost function is then $C = C_n(m)$ where n is the number of layers and m is the number of voxels in the vertical column.

The layer configuration that minimizes C can be computed with dynamic programming. In order for this to be possible, the problem must exhibit *optimal substructure* and *overlapping subproblems* (Cormen et al., 2001). The problem has optimal substructure because of the independence between non-adjacent layers, i.e. an optimal configuration of layers $1 \dots i-1$ will still be optimal regardless of the position of layer i. (As in C_k, we consider only the voxels below the layer.) The overlapping subproblems occur since computing the optimal position of any layer greater than i requires computing the optimal configuration of layers $1 \dots i$. Therefore, the optimal configuration can be solved with dynamic programming. The recursive formulation in Equation 7.14 lends itself easily to the table method, and the solution can extracted by backtracking. See Figure 7.3 for an example.

Many parts of the heightmap will not need all n layers. The extra layers will be free to fit the noise in the measurements. To avoid this, we incorporate the Bayesian Information

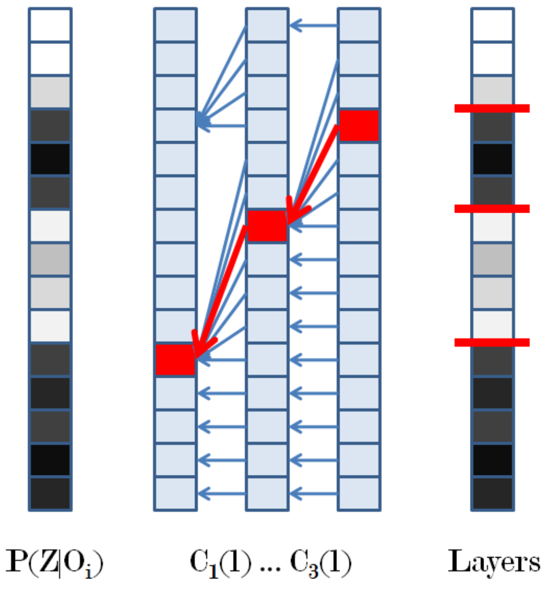


Figure 7.3: The optimial layer configuration is computed with dynamic programming.

Criterion (BIC):

$$C_{BIC} \quad = \quad -\ln\, P(Z|L)P(L) + \frac{1}{2}n\, \ln\, |Z_x| \tag{7.17}$$

where $|Z_x|$ is the number of measurements interacting with the heightmap pixel x. The first part of the equation is exactly C and the second part adds a penalty of $\ln\, |Z_x|$ for every layer in the model. We can add this penalty into our recursive formulation by adding $\ln\, |Z_x|$ at each layer unless the layer position is the same as the preceding layer.

$$C_k^{BIC}(l) \quad = \quad \begin{cases} I_{l'}^{l} + C_{k-1}(l') + T(l \neq l')\frac{1}{2}\ln\, |Z_x| & \text{if } odd(k) \\ \bar{I}_{l'}^{l} + C_{k-1}(l') + T(l \neq l')\frac{1}{2}\ln\, |Z_x| & \text{if } even(k) \end{cases} \tag{7.18}$$

(7.19)

Thus model selection is performed by preferring layers to *collapse* unless there is sufficient evidence to support them. The table required to solve the problem is of size $m \times n$, and the sum variables are of size m. Therefore the algorithm takes $O(mn)$ time and space per heightmap pixel, and the whole $w \times h$ heightmap takes $O(whmn)$ time and $O(wh + mn)$ space.

We have implemented the heightmap algorithm on the GPU using NVIDIA's CUDA platform (Nvidia). While our method can be parallelized using any technology, we found CUDA to have a number of advantages. First, CUDA provides shared memory which multiple threads can access. We use this to our advantage by having a different thread compute the occupancy likelihood for each voxel in a vertical column, writing the results to an array in shared memory. After the likelihood computation, one of the threads performs the dynamic programming computation. Second, CUDA allows for divergent branching at the block level. (Each block is composed of multiple threads, see NVIDIA's CUDA (Nvidia) for details.) This allows for greater efficiency when all cells assigned to a block have been masked out due to overlapping the previous heightmap in a video sequence (See Section 7.3.1). In that case, the block of threads can terminate quickly, freeing up resources for the next block. Figure 7.5b shows the blocks of a heightmap layout that can take advantage of this early exit divergence.

7.3 3D Reconstruction with Heightmaps

We will now describe how to use the heightmap method to reconstruct scenes from video or photo collections. Reconstruction consists of the following steps:

1. Lay out the volume of interest.
2. Compute the n-layer heightmap.
3. Extract the surface mesh and generate texture maps.

Step 2 was already described in Section 7.2.

7.3.1 Layout

The volume of interest for heightmap computation is defined by its position, orientation, size, and resolution. Note that the x and y resolution of the heightmap directly influence the memory consumption of our computation, but the z (height) resolution does not (except for temporary local storage). The heightmap computation assumes that the vertical

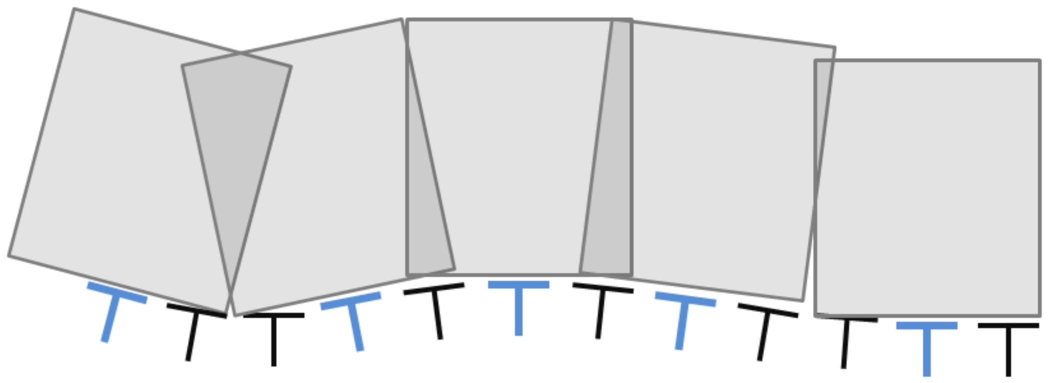


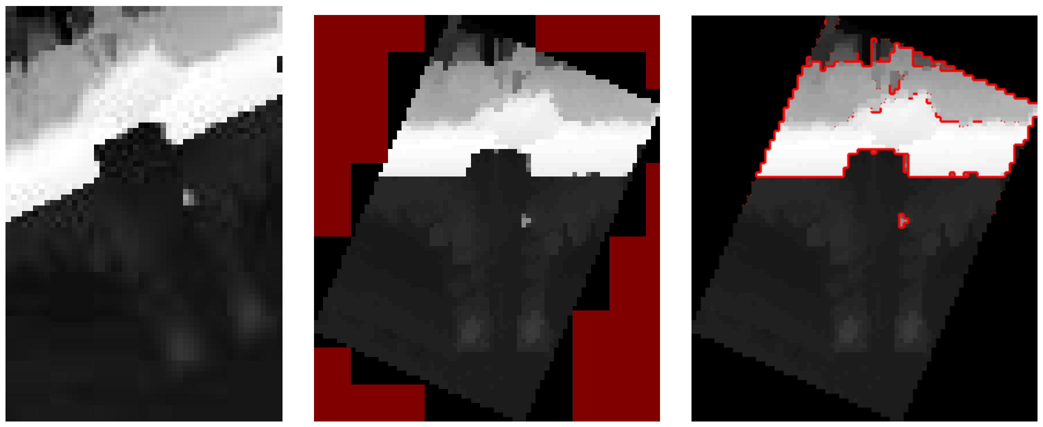
Figure 7.4: Volume of interest layout for processing video. Each volume is processed independently.

direction is known. Besides that constraint, the volume of interest can be defined arbitrarily. For processing large datasets like video of an entire street, it makes sense to define several volumes of interest and process them independently. For video, a frame is chosen as reference, and the volume of interest is defined with respect to the camera's coordinate system for that frame. Reference frames are chosen at irregular intervals where the spacing is determined by overlap with the previous volume. See Figure 7.4 for an example.

For photo collections, the views are unordered so heightmaps cannot be laid out in sequence. Some photo collections are small enough that a single heightmap can be used, but usually there are too many views and too much area. Instead, the area occupied by the 3D points (from SfM) is broken up into tiles depending on the desired resolution of the reconstruction. Only tiles with a reasonable number of 3D points in them are computed.

Computing the vertical direction is essential for the heightmap method. For video captured with the Urbanscape system, the vertical direction is given by the GPS/INS measurements. Otherwise it can be computed by examining vanishing points as described in Section 4.4.1. For photo collections, the vertical direction can be found using a heuristic derived from photography practices. Most photographers will tilt the camera, but not allow it to roll. In other words, the x axis of the camera stays perpendicular to gravity. This heuristic can be used to compute the vertical direction as a homogeneous least squares problem as shown by Szeliski (2005).

Facades and other vertical surfaces will appear as discontinuities in our heightmap. To



(a) (b) (c)

Figure 7.5: (a) Heightmap computed without axis alignment. (b) Heightmap computed with axes aligned to dominant surface normal. Dark red indicates complete blocks where a CUDA kernel can diverge with an early exit. (c) Bright red indicates detected heightmap discontinuities.

avoid staircase discontinuities due to the pixel grid, we align the grid's x and y axes to the facade's surface normal. This can be done using the method in Section 4.4.1. See Figure 7.5a-b.

7.3.2 Mesh Extraction and Texturing

Once a heightmap is computed, the next step is to create a polygonal mesh. This could be done easily by generating a quadrilateral mesh for each heightmap layer. However, we wish to pay special attention to large changes in height since they usually represent vertical walls. We can detect these height discontinuities by thresholding the height gradient magnitude. For every discontinuity between two pixels, a vertical quadrilateral is generated between full and empty space. This can be done by sorting all the layer heights from both pixels and considering them one pair at a time. Unless the height values in a pair are equal, a vertical surface must be generated.

Now that the surface geometry has been generated, the next step is to compute the texture map for the mesh. In previous chapters, the surfaces were generated from depthmaps, so the geometry and the color were determined from the same view. In that case, texture mapping was simply a matter of back-projecting the image onto the surface. In this case, the surface is composed from many views, some of them occluded. In the case of photo collections these views have very different appearances. Therefore, we compute the texture map by combining the color information from all views with a median method. For each surface point, the color is determined by projecting that point into all the images and accumulating an intensity histogram for each color channel. The histogram vote is down-weighted if the depthmap for that view differs from the depth of the reconstructed surface indicating an occlusion. The final color is then computed simply as the median value of the histogram for each color channel.

7.4 Experiments

We have tested our n-layer heightmap method on street-level video from Urbanscape and photo collections downloaded from the web. For the video datasets, the depthmaps were computed with the plane-sweeping stereo method in Chapter 4. To compute the camera poses for the photo collections, we used the method of Frahm et al. (2010). The output of their system also gives a clustering of the images which can be used to select compatible views for stereo. We computed a depthmap for each photograph by selecting the 20 views in the same cluster with the most matched and triangulated SIFT points in common. Because views in a cluster have similar viewpoint and appearance, the simple plane-sweep method can be used. However, NCC is used instead of the SAD matching score, since there is still some appearance variation between images. Results are shown in Figure 7.6.

Figures 7.7 and 7.8 show 3D heightmap reconstructions before and after texture mapping. Figure 7.9 shows the improvement gained by using multiple layers in the heightmap. Overhanging structures are recovered while the clean and compact nature of the reconstruction is preserved. Figures 7.10 show the results of the reconstructions from video. Figures 7.11 show the results of the reconstructions from photo collections. Most reconstructions required only 3 layers. 5 layers were used for some models, and the model of the Roman Coliseum used 7 layers. Note that even though the model selection adapts to the number of layers in the scene, it is more efficient to specify fewer layers if possible.

Our system can process video at 13.33 Hz. Computing a 3-layer 100x100 heightmap with 100 height levels from 48 depthmaps takes only 69 ms on the GPU. The other steps are not as fast as we did not focus as much on optimizing them. Converting the heightmap into a mesh takes 609 ms, and generating texture maps takes 1.57 seconds. The total time for processing a heightmap is 2.25 seconds. However, heightmaps only need to be computed about every 30 frames of video. Therefore, our system can process video at 13.33 frames per second. Reconstructing photo collections is more challenging. Each scene takes 20-30 minutes, and most of that time is spent computing NCC stereo.

7.5 Discussion

We have proposed a novel n-layer heightmap method for depthmap fusion and surface generation. The structure of urban scenes allows a multi-layer heightmap to model most aspects of the scene, and the heightmap is ideal for reconstructing vertical walls since the model does not support noise variations perpendicular to the wall. One of the main advantages of this approach is that it is highly parallel and memory efficient and can therefore be implemented on the GPU. This efficiency comes at a cost, since not all the detail of a scene can be captured with a heightmap.

Experiments have shown our heightmap method to be effective at reconstructing a wide variety of urban scenes both from street-level video and from Internet photo collections. As is expected, the resulting 3D models do not capture all the detail of the scene, but they are clean, compact, and the surface is continuous without holes. One of the weaknesses at this point is the texture maps. The simple method used blends color information from many views. This inevitably results in a loss of resolution since not all the views are perfectly calibrated, the surface geometry is not perfect, and the images have all been sampled differently. Thus the images back-projected onto the surface will be slightly misaligned, so blending results in low-pass filtering. To obtain higher quality texture maps, an image mosaicking approach such as Gracias et al. (2009) could be used.

Another direction for future work would be to incorporate spatial regularization to enforce smoothness between neighboring height pixels. While we have shown that regularization is not necessary to obtain a good result, it could help in some cases where too few depth measurements lead to noisy results.

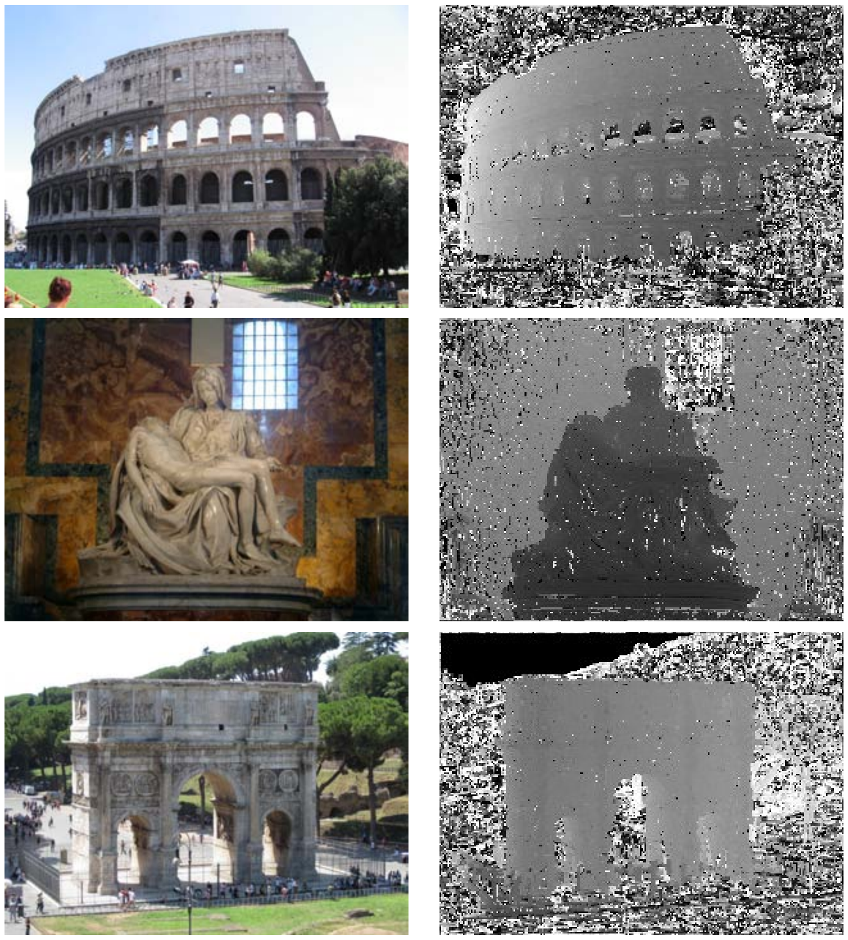


Figure 7.6: Original photos and depthmaps computed from Internet photo collections.

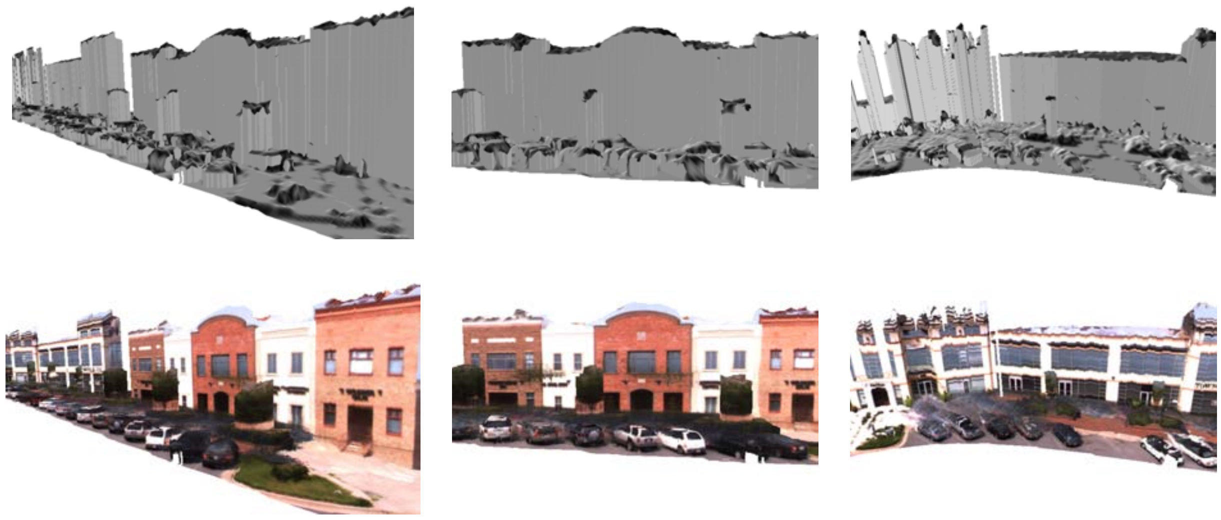


Figure 7.7: Untextured and textured 3D models produced by our system. This challenging scene features many reflective cars and glass store-front facades.

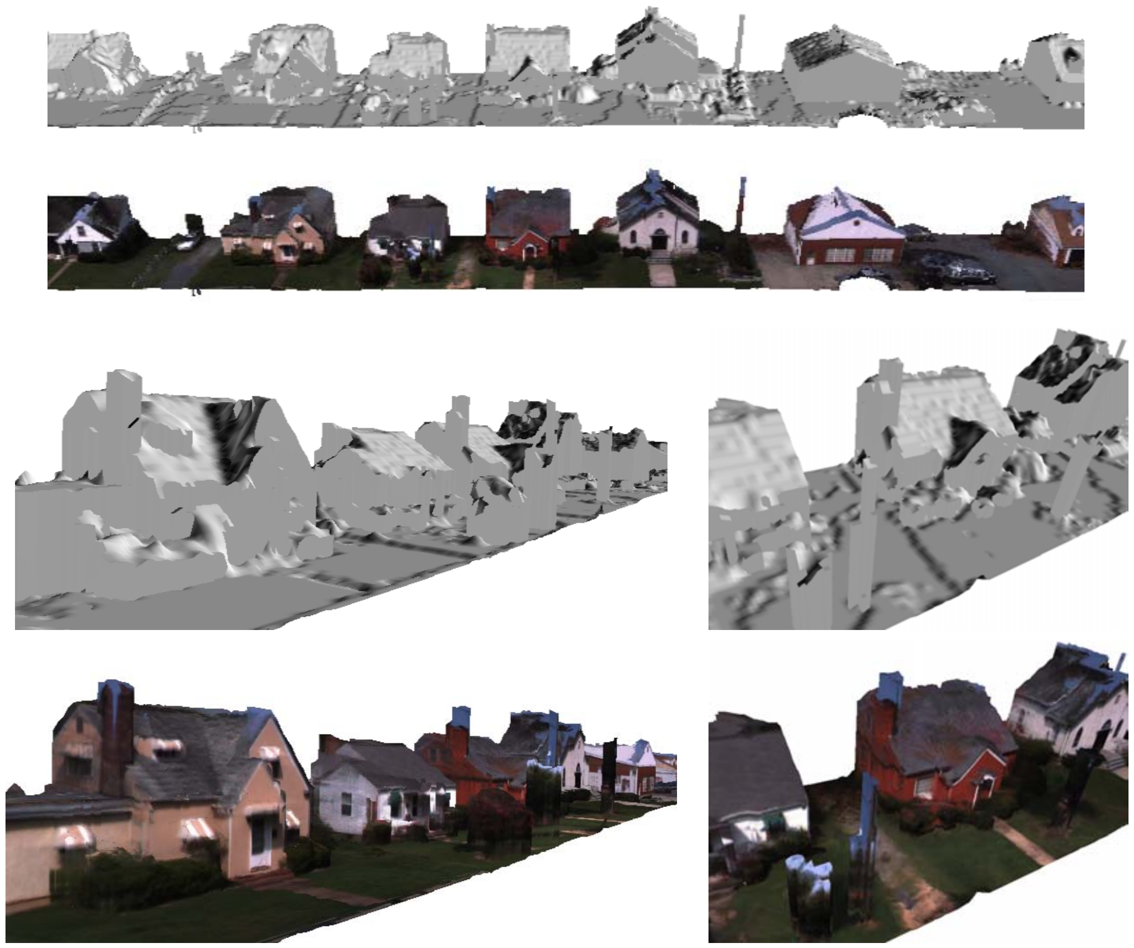


Figure 7.8: Untextured and textured 3D models of a residential scene reconstructed by our algorithm.

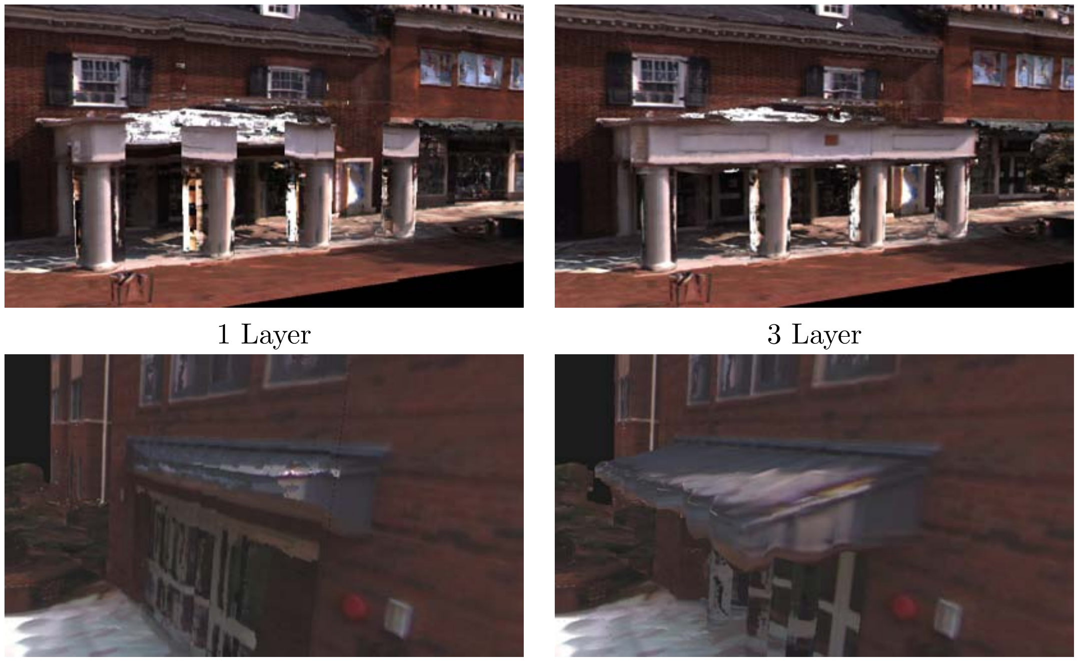


Figure 7.9: 1-layer and 3-layer reconstructions.

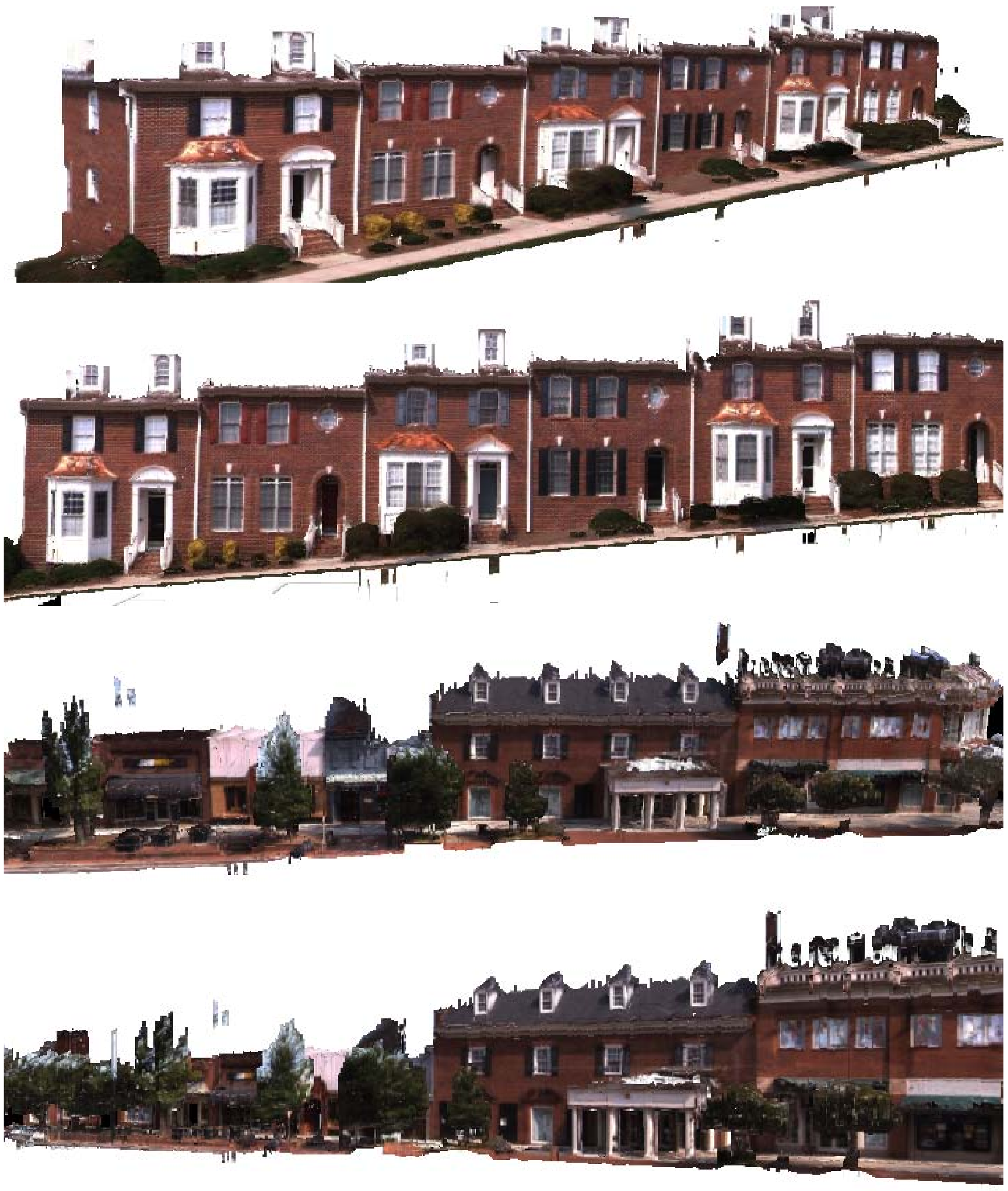


Figure 7.10: More 3D reconstructions from video.

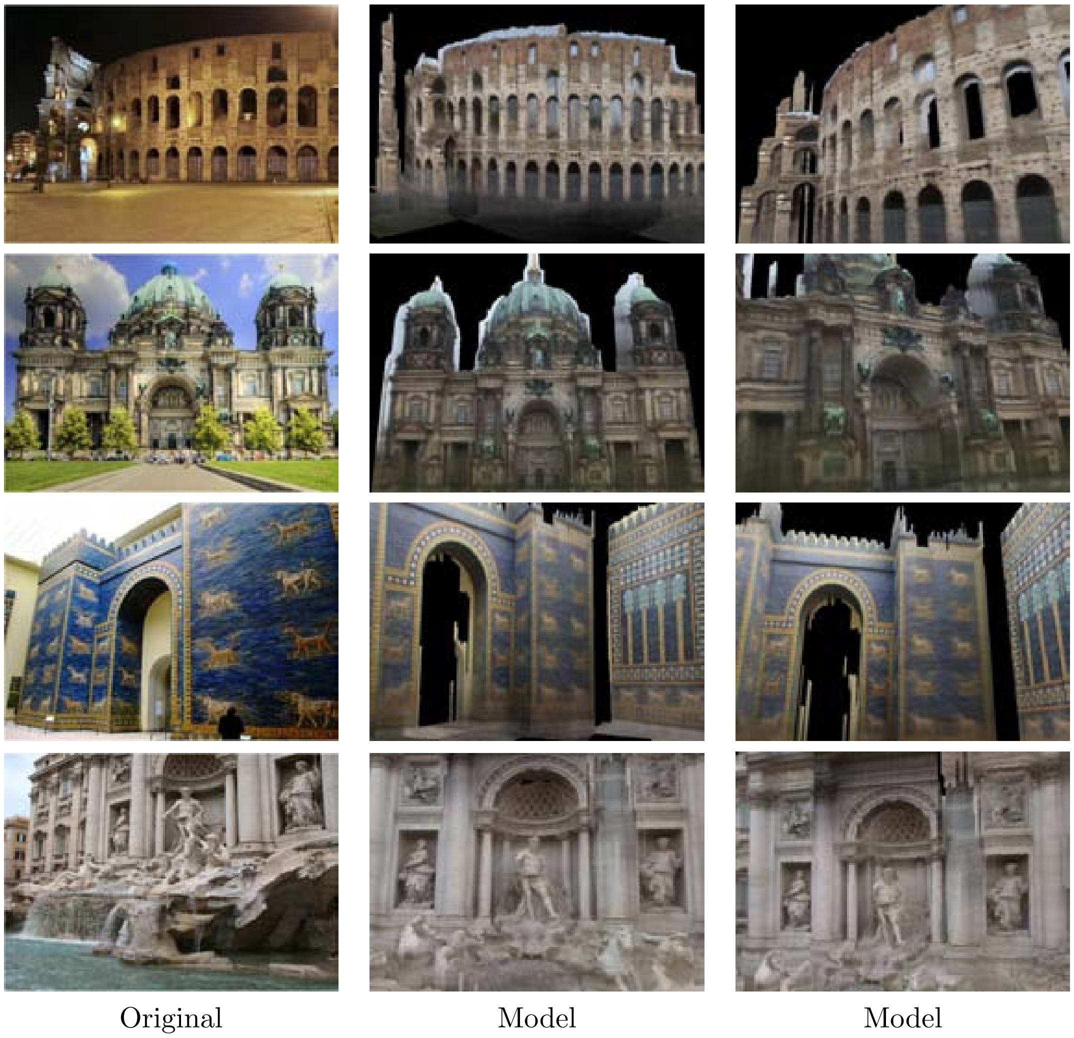


Figure 7.11: 3D reconstructions from Internet photo collections.

CHAPTER 8

Conclusion

8.1 Summary

I will now discuss how this dissertation has supported the claims of the thesis statement:

> Dense 3D reconstruction of large-scale urban environments can be performed automatically from video captured from street-level. By using parallelizable and scalable algorithms which take advantage of urban scene structure, the 3D scene can be reconstructed accurately and efficiently, even in real-time.

The thesis statement highlights the main challenge of this work: scale. Reconstructing cities from ground-level requires processing an enormous amount of data. Algorithms must be scalable. Processing speed, memory usage, and output model size are all challenges derived from the scale of the problem. Chapter 4 presents a multi-sweep stereo that runs in real-time on the GPU. This allows depthmaps to be computed for every frame, and these then serve as inputs to the other methods in Chapters 6 and 7. Chapter 5 discusses a variable baseline/resolution view selection method for stereo that achieves a target depth resolution in $O(z^3)$ time compared to $O(z^6)$ time for the fixed-baseline approach. The method is particularly useful for street-level imagery for which the scene exhibits a large depth range. The piecewise-planar stereo in Chapter 6, although not real-time, is scalable. All images can be processed independently, and the reconstruction is still globally consistent since plane hypotheses are linked over multiple views. Finally, Chapter 7 uses a heightmap representation to perform volumetric fusion in real-time and with less memory. The simplified scene can be stored compactly as a heightmap rather than a 3D mesh.

Accuracy is another goal of the thesis. Using multiple sweeping directions in Chapter 4 resulted in more than twice as precise depth measurements compared to a single plane-sweep. Chapter 5 addresses the geometric factor of depth measurement precision, and shows how varying the baseline and resolution appropriately can yield constant depth resolution over the depth range (compared to fixed baseline stereo where depth resolution declines quadratically with depth). Experimental results on ground truth show this theory to hold. Chapter 6 looks for planes in the scene to improve the accuracy of the reconstruction. The heightmap used in Chapter 7 cannot capture all the detail of the scene but it ensures walls are vertical thereby improving accuracy when this is the case.

The thesis also claims that these algorithms achieve their performance by taking advantage of urban scene structure. The plane-sweep method of Chapter 4 finds the principal surface normals in a planar urban scene by using orthogonality, verticality, and camera motion. Furthermore, the planarity of the scene allows the algorithm to construct a discriminative prior on the location of planes in the scene. Chapter 6 identifies planar regions by looking at the texture regularity and color of the scene. Chapter 7 shows that urban scenes can effectively be modeled with heightmaps which rely on and enforce verticality.

The methods presented in this dissertation address each of three major aspects of dense 3D reconstruction, namely multi-view correspondence, triangulation geometry, and surface generation. Chapter 4 computes more accurate correspondences by aligning the plane-sweep directions to the surfaces in the scene. Chapter 5 varies the baseline and resolution to specifically deal with the *geometric factor* of depth precision. Chapter 6 selectively enforces planar surfaces which improves accuracy, and Chapter 7 uses a heightmap representation to model the scenes surfaces.

8.2 Future Work

For future work, I consider the following directions and ideas:

Line Features Line features are an important part of urban scene structure that we have so far overlooked. Similar to texture regularity, lines indicate planar surfaces, and are important features to reconstruct correctly. These high frequency edges are quite salient, and so it is important that they remain lines in the 3D reconstruction. Converging line features could also identify vanishing points which could identify the urban scene normals. In the piecewise planar reconstruction, lines could be used in the planar surface classifier or 3D lines could be used in the data term to favor planes that intersect them as in Sinha et al. (2009). Line features could be particularly helpful for reconstructing indoor environments where textureless walls are prevalent.

Improved Classification and Reconstruction Finding planes based on manmade surface appearance is a promising direction. This concept could be expanded to handle other types of geometric primitives such as cylinders, spheres, and cones, similar to Labatut et al. (2009), or b-splines similar to Bleyer et al. (2010). These primitives would allow a much wider range of surfaces to be reconstructed with simple shapes. This simplification not only reduces model size but also produces a cleaner, more accurate reconstruction. In addition to more primitives, more classes could be considered. Knowing the surface type allows the appropriate reconstruction technique to be used, or at the very least allows the method's parameters to adapt to the surface. For example, pixels classified as glass window should not be matched using typical matching scores since the glass surface's reflections can change the appearance significantly from view to view. Instead, a matching score based on edges that seeks to identify the window's frame might perform better for this type of surface. Reconstruction should be able to improve classification as well. Scene parsing approaches like Tighe et al. (2010) could be applied and updated to take into account 3D features like curvature, height above the ground, absolute scale, and object depth boundaries. The relationship between reconstruction and recognition problems has been shown to be important by Cornelis et al. (2008).

Heightmap Reconstruction Including regularization between neighboring heightmap pixels could prove helpful in cases where little depth data is available. One approach could be to smooth or denoise each layer after the initial computation. But this would not allow large errors to be corrected. Adding regularization into the multi-layer computation is difficult however. Adding dependence between neighboring layers would mean that dynamic programming could no longer solve the problem. Graph-cut methods which are typically used for these image optimization problems could not handle multiple labels per pixel. Using graph-cuts to update one layer at a time will also not work because each layer would easily get stuck in local minima. Moving two layers together will avoid this, but the set of possible locations would become quadratic. As an approximation, one could compute the data term for each layer as if all other layers were in their optimal positon disregarding smoothness penalties. This would be a lower bound approximation to the true data term, but it would allow a single layer to move because the data term would behave as though the other layers were moving to reduce the energy as well.

Besides regularization, other enhancements could be made to the method. For example, texture maps could be generated using a texture mosaicing approach like Gracias et al. (2009). Currently the texture maps lose quality because all views are blended together. All the misalignments due to geometry error and camera calibration error act as a low-pass filter when blending. Computing a mosaic would attempt to select a single crisp image to texture the region on the surface.

Finally, because the heightmap uses the probabilistic occupancy grid framework, it should be trivial to fuse data from different modalities, provided that their noise characteristics are known. Fusing aerial-based stereo or LiDAR measurements would result in greater completeness, especially for the roof-tops which are not observed from the ground.

Completing the System At this point, not all the methods presented in each chapter work together as a complete system. Currently the variable baseline/resolution view selection strategy only works for fronto-parallel sweeps. Handling non-fronto-parallel planes is a non-trivial extension to the current algorithm. The main reason is that for non-fronto-parallel planes, each point on the plane has a different depth, and so baseline and image resolution must be selected per pixel. This complicates the GPU implementation considerably. Selection of image resolution has been implemented with an image scale pyramid (called a mip-map in graphics processing), and until recently mip-map levels were determined automatically and could not be selected according to depth as is required by our algorithm. If this were implemented, the variable baseline/resolution approach could be used both for the multi-sweep stereo and to compute the matching scores for the plane hypothesis in the piecewise planar reconstruction. This would allow these methods to reconstruct the scene at a greater distance and would reduce computation in the near range.

BIBLIOGRAPHY

Agarwal, S., Snavely, N., Simon, I., Seitz, S. M., and Szeliski, R. (2009). Building rome in a day. In *ICCV*.

Baillard, C., Schmid, C., Zisserman, A., and Fitzgibbon, A. (1999). Automatic line matching and 3d reconstruction of buildings from multiple views. In *In ISPRS Conference on Automatic Extraction of GIS Objects from Digital Imagery*.

Belhumeur, P. N. (1996). A bayesian approach to binocular stereopsis. *International Journal of Computer Vision (IJCV)*.

Besl, P. J. and McKay, N. D. (1992). A method for registration of 3-d shapes. *IEEE Transactions on Acoustics, Speech and Signal Processing (PAMI)*, 14.

Birchfield, S. and Tomasi, C. (1998). A pixel dissimilarity measure that is insensitive to image sampling. *Pattern Analysis and Machine Intelligence (PAMI)*.

Birchfield, S. and Tomasi, C. (1999a). Depth discontinuities by pixel-to-pixel stereo. *International Journal of Computer Vision (IJ*, 35:269–293.

Birchfield, S. and Tomasi, C. (1999b). Multiway cut for stereo and motion with slanted surfaces. In *Int. Conf. on Computer Vision (ICCV)*, pages 489–495.

Bleyer, M., Rother, C., and Kohli, P. (2010). Surface stereo with soft segmentation. In *Computer Vision and Pattern Recognition (CVPR)*.

Bosse, M., Rikoski, R., Leonard, J., and Teller, S. (2003). Vanishing points and 3d lines from omnidirectional video. *The Visual Computer*, 19(6):417–430.

Boykov, Y. and Kolmogorov, V. (2004). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *Pattern Analysis and Machine Intelligence (PAMI)*.

Boykov, Y., Veksler, O., and Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239.

Burt, P., Wixson, L., and Salgian, G. (1995). Electronically directed "focal" stereo. In *Int. Conf. on Computer Vision (ICCV)*, pages 94–101. IEEE Computer Society.

Chen, G., Kua, J., Shum, S., Naikal, N., Carlberg, M., and Zakhor, A. (2010). Indoor localization algorithms for a human-operated backpack system. In *Int. Symp. on 3D Data, Processing, Visualization and Transmission (3DPVT)*.

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

Coorg, S. and Teller, S. (1999). Extracting textured vertical facades from controlled close-range imagery. In *Computer Vision and Pattern Recognition (CVPR)*.

Cormen, T. H., Leisorson, C. E., Rivest, R. L., and Stein, C. (2001). *Introduction to Algorithms*. The MIT Press.

Cornelis, N., Leibe, B., Cornelis, K., and Gool, L. V. (2008). 3d urban scene modeling integrating recognition and reconstruction. *International Journal of Computer Vision (IJCV).*

Debevec, P., Taylor, C., and Malik, J. (1996). Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *SIGGRAPH*, pages 11–20.

Dick, A. R., Torr, P. H. S., Ruffle, S. J., and Cipolla, R. (2001). Combining single view recognition and multiple view stereo for architectural scenes. In *International Conference on Computer Vision (ICCV).*

Dyer, C. R. (2001). Volumetric scene reconstruction from multiple views. *Foundations of Image Analysis*, pages 469–489.

Fabio, R. (2003). From point cloud to surface: The modeling and visualization problem. In *International Workshop on Visualization and Animation of Reality-based 3D Models.*

Falkenhagen, L. (1997). Hierarchical block-based disparity estimation considering neighbourhood constraints. In *Int. workshop on SNHC and 3D Imaging.*

Felzenszwalb, P. F. and Huttenlocher, D. P. (2004a). Efficient belief propagation for early vision. In *Compute Vision and Pattern Recognition (CVPR).*

Felzenszwalb, P. F. and Huttenlocher, D. P. (2004b). Efficient graph-based image segmentation. *International Journal of Computer Vision (IJCV).*

Förstner, W. (1999). 3d-city models: Automatic and semiautomatic acquisition methods. Proc. Photogrammetric Week, Univ. of Stuttgart, Inst. for Photogrammetry. 291-303.

Forsyth, D. A. and Ponce, J. (2002). *Computer Vision: A Modern Approach.* Prentice Hall.

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

Früh, C. and Zakhor, A. (2003). Constructing 3d city models by merging aerial and ground views. *IEEE Computer Graphics and Applications*, 23:52–61.

Früh, C. and Zakhor, A. (2004). An automated method for large-scale, ground-based city model acquisition. *International Journal of Computer Vision (IJCV).*

Furukawa, Y., Curless, B., Seitz, S. M., , and Szeliski, R. (2009a). Manhattan-world stereo. In *Computer Vision and Pattern Recognition (CVPR).*

Furukawa, Y., Curless, B., Seitz, S. M., , and Szeliski, R. (2009b). Reconstructing building interiors from images. In *International Conference on Computer Vision (ICCV).*

Furukawa, Y., Curless, B., Seitz, S. M., and Szeliski, R. (2010). Towards internet-scale multi-view stereo. In *CVPR.*

Furukawa, Y. and Ponce, J. (2008). Accurate, dense, and robust multi-view stereopsis. *Pattern Analysis and Machine Intelligence (PAMI).*

Gallup, D., Frahm, J.-M., Mordohai, P., and Pollefeys, M. (2008). Variable baseline/resolution stereo. In *Computer Vision and Pattern Recognition (CVPR)*.

Gallup, D., Frahm, J.-M., Mordohai, P., Qingxiong, Y., and Pollefeys, M. (2007). Real-time plane-sweeping stereo with multiple sweeping directions. In *Computer Vision and Pattern Recognition (CVPR)*.

Gallup, D., Frahm, J.-M., and Pollefeys, M. (2009). Real-time depth boundary optimization for local area-based stereo. In *3DSM*.

Gallup, D., Frahm, J.-M., and Pollefeys, M. (2010a). A heightmap model for efficient 3d reconstruction from street-level video. In *3DPVT*.

Gallup, D., Frahm, J.-M., and Pollefeys, M. (2010b). Piecewise planar and non-planar stereo for urban scene reconstruction. In *CVPR*.

Gallup, D., Pollefeys, M., and Frahm, J.-M. (2010c). 3d reconstruction using an n-layer heightmap. In *DAGM*.

Geyer, C., Templeton, T., Meingast, M., , and Sastry, S. S. (2006). The recursive multi-frame planar parallax algorithm. In *3DPVT*.

Goesele, M., Snavely, N., Curless, B., Hoppe, H., and Seitz, S. M. (2007). Multi-view stereo for community photo collections. In *International Conference on Computer Vision (ICCV)*.

Gracias, N., Mahoor, M., Negahdaripour, S., and Gleason, A. (2009). Fast image blending using watersheds and graph cuts. *Image and Vision Computing*, 27.

Guan, L., Franco, J.-S., and Pollefeys, M. (2008). 3d object reconstruction with heterogeneous sensor data. In *Int. Symp. on 3D Data, Processing, Visualization and Transmission (3DPVT)*.

Gulch, E., Muller, H., and Labe, T. (1999). Integration of automatic processes into semi-automatic building extraction. *IAP*, 32.

Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *Fourth Alvey Vision Conference*.

Hartley, R. and Zisserman, A. (2000). *Multiple View Geometry in Computer Vision*. Cambridge University Press.

Hirschmuller, H. (2008). Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 30(2):328–341.

Hirschmüller, H., Innocent, P. R., and Garibaldi, J. (2002). Real-time correlation-based stereo vision with reduced border errors. *Int. J. Comput. Vision*, 47(1-3):229–246.

Hirschmuller, H. and Scharstein, D. (2008). Evaluation of stereo matching costs on images with radiometric differences. *Pattern Analysis and Machine Intelligence (PAMI)*.

Hoiem, D., Efros, A., and Hebert, M. (2006). Putting objects in perspective. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, pages 2137–2144.

Hoiem, D., Efros, A. A., and Hebert, M. (2005). Geometric context from a single image. In *International Conference on Computer Vision (ICCV)*.

Irschara, A., Zach, C., and Bischof, H. (2007). Towards wiki-based dense city modeling. In *VRML Workshop in conjunction with ICCV 2007*.

Kang, S., Szeliski, R., and Chai, J. (2001). Handling occlusions in dense multi-view stereo. In *Computer Vision and Pattern Recognition (CVPR)*, pages I:103–110.

Kazhdan, M., Bolitho, M., and Hoppe, H. (2006). Poisson surface reconstruction. In *Eurographics Symposium on Geometry Processing*.

Koch, R., Pollefeys, M., and Gool, L. V. (1998). Multi viewpoint stereo from uncalibrated video sequences. In *European Conference on Computer Vision (ECCV)*.

Kolmogorov, V. and Zabih, R. (2002). What energy functions can be minimized via graph cuts? *Pattern Analysis and Machine Intelligence (PAMI)*.

Kumar, S. and Hebert, M. (2003). Discriminative random fields: A discriminative framework for contextual interaction in classification. In *International Conference on Computer Vision (ICCV)*.

Kutulakos, K. N. and Seitz, S. M. (2000). A theory of shape by space carving. *International Journal of Computer Vision*.

Labatut, P., Pons, J.-P., and Keriven, R. (2009). Hierarchical shape-based surface reconstruction for dense multi-view stereo. In *3-D Digital Imaging and Modeling (3DIM)*.

Margaritis, D. and Thrun, S. (1998). Learning to locate an object in 3d space from a sequence of camera images. In *Proc. of Int. Conf. on Machine Learning (ICML)*.

Matthies, L., Szeliski, R., and Kanade, T. (1989). Kalman filter-based algorithms for estimating depth from image sequences. *IJCV*.

Micusik, B. and Kosecka, J. (2009). Piecewise planar city 3d modeling from street view panoramic sequences. In *Computer Vision and Pattern Recognition (CVPR)*.

Morgan, M. and Tempfli, K. (2000). Automatic building extraction from airborne laser scanning data. *Proc. 19th Intl Soc. Photogrammetry and Remote Sensing Congress (ISPRS),*.

Müller, P., Zeng, G., Wonka, P., and V., G. L. (2007). Image-based procedural modeling of facades. *SIGGRAPH*.

Nvidia. Cuda. `http://www.nvidia.com/cuda`.

Ogale, A. and Aloimonos, Y. (2004). Stereo correspondence with slanted surfaces: Critical implications of horizontal slant. In *Computer Vision and Pattern Recognition (CVPR)*, pages I: 568–573.

Okutomi, M. and Kanade, T. (1993). A multiple-baseline stereo. *Pattern Analysis and Machine Intelligence (PAMI)*.

Pathak, K., Birk, A., Poppinga, J., and Schwertfeger, S. (2007). 3d forward sensor modeling and application to occupancy grid based sensor fusion. In *International Conference on Intelligent RObots and Systems (IROS)*.

Pock, T., Schoenemann, T., Graber, G., Bischof, H., and Cremers, D. (2008). A convex formulation of continuous multi-label problems. In *European Conference on Computer Vision (ECCV)*.

Pollefeys, M., Gool, L. V., Vergauwen, M., Verbiest, F., Cornelis, K., Tops, J., and Koch, R. (2004). Visual modeling with a hand-held camera. *International Journal of Computer Vision (IJCV)*.

Pollefeys, M., Nister, D., Frahm, J.-M., Akbarzadeh, A., Mordohai, P., Clipp, B., Engels, C., Gallup, D., Kim, S.-J., Merrell, P., Salmi, C., Sinha, S., Talton, B., Wang, L., Yang, Q., Stewenius, H., Yang, R., Welch, G., and Towles, H. (2008). Detailed real-time urban 3d reconstruction from video. *International Journal of Computer Vision (IJCV)*.

Raguram, R., Frahm, J.-M., and Pollefeys, M. (2008). A comparative analysis of ransac techniques leading to adaptive real-time random sample consensus. In *European Conference on Computer Vision (ECCV)*.

Ren, X. and Malik, J. (2003). Learning a classification model for segmentation. In *International Conference on Computer Vision (ICCV)*.

Sato, T., Kanbara, M., Yokoya, N., and Takemura, H. (2002). Dense 3-d reconstruction of an outdoor scene by hundreds-baseline stereo using a hand-held video camera. *International Journal of Computer Vision (IJCV)*.

Saurer, O., Fraundorfer, F., and Pollefeys, M. (2010). Omnitour: Semi-automatic generation of interactive virtual tours from omnidirectional video. In *Int. Symp. on 3D Data, Processing, Visualization and Transmission (3DPVT)*.

Saxena, A., Chung, S. H., and Ng, A. Y. (2007). 3-d depth reconstruction from a single still image. *International Journal of Computer Vision (IJCV)*.

Scharstein, D. and Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision (IJCV)*.

Scharstein, D. and Szeliski, R. (2003). High-accuracy stereo depth maps using structured light. In *Computer Vision and Pattern Recognition (CVPR)*.

Schindler, G. and Dellaert, F. (2004). Atlanta world: an expectation maximization framework for simultaneous low-level edge grouping and camera calibration in complex man-made environments. In *Computer Vision and Pattern Recognition (CVPR)*, pages 203–209.

Schindler, G. and Dellaert, F. (2010). Probabilistic temporal inference on reconstructed 3d scenes. In *Computer Vision and Pattern Recognition (CVPR)*.

Schindler, G., Krishnamurthy, P., and Dellaert, F. (2006). Line-based structure from motion for urban environments. In *Int. Symp. on 3D Data, Processing, Visualization and Transmission (3DPVT).*

Schuon, S., Theobalt, C., Davis, J., and Thrun, S. (2008). High-quality scanning using time-of-flight depth superresolution. In *Computer Vision and Pattern Recognition (CVPR).*

Seitz, S., Curless, B., Diebel, J., Scharstein, D., and Szeliski, R. (2006). A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Computer Vision and Pattern Recognition (CVPR).*

Shi, J. and Tomasi, C. (1994). Good features to track. In *Computer Vision and Pattern Recognition (CVPR).*

Sinha, S. N., Steedly, D., and Szeliski, R. (2009). Piecewise planar stereo for image-based rendering. In *International Conference on Computer Vision (ICCV).*

Snavely, N., Seitz, S. M., and Szeliski, R. (2006). Photo tourism: Exploring photo collections in 3d. In *SIGGRAPH*, pages 835–846.

Sun, C. (2002). Fast stereo matching using rectangular subregioning and 3d maximum-surface techniques. *International Journal of Computer Vision (IJCV).*

Szeliski, R. (2005). Image alignment and stitching: A tutorial. In *Microsoft Research Technical Report.*

Szeliski, R. and Scharstein, D. (2004). Sampling the disparity space image. *Pattern Analysis and Machine Intelligence (PAMI).*

Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., and Rother, C. (2006). A comparative study of energy minimization methods for markov random fields. In *European Conference on Computer Vision (ECCV).*

Teller, S. (1998). Automated urban model acquisition: Project rationale and status. In *Image Understanding Workshop*, pages 455–462.

Teller, S., Antone, M., Bodnar, Z., Bosse, M., Coorg, S., Jethwa, M., and Master, N. (2003). Calibrated, registered images of an extended urban area. *Int. Journal of Computer Vision (IJCV)*, 53(1):93–107.

Tighe, J. and Lazebnik, S. (2010). Superparsing: Scalable nonparametric image parsing with superpixels. In *European Conference on Computer Vision (ECCV).*

Torr, P. H. S. and Zisserman, A. (2000). Mlesac: a new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding (CVIU).*

Vanegas, C. A., Aliaga, D. G., and Benes, B. (2010). Building reconstruction using manhattan-world grammars. In *Computer Vision and Pattern Recognition (CVPR).*

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

Wang, X., Totaro, S., Taillandier, F., Hanson, A., , and Teller, S. (2002). Recovering facade texture and microstructure from real-world images. In *Proc. 2nd International Workshop on Texture Analysis and Synthesis (ECCV Workshop).*

Werner, T. and Zisserman, A. (2002). New techniques for automated architectural reconstruction from photographs. In *European Conf. on Computer Vision (ECCV)*, pages 541–555.

Xiao, J., Fang, T., Tan, P., Zhao, P., Ofek, E., and Quan, L. (2008). Image-based facade modeling. *SIGGRAPH Asia*.

Xiao, J., Fang, T., Zhao, P., Lhuillier, M., and Quan, L. (2009). Image-based street-side city modeling. In *SIGGRAPH Asia*.

Xiao, J. and Quan, L. (2009). Multiple view semantic segmentation for street view images. *International Conference on Computer Vision (ICCV).*

Yang, Q., Wang, L., Yang, R., Wang, S., Liao, M., and Nister, D. (2006). Real-time global stereo matching using hierarchical belief propagation. In *The British Machine Vision Conference (BMVC).*

Yang, R. and Pollefeys, M. (2003). Multi-resolution real-time stereo on commodity graphics hardware. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages I: 211–217.

Yang, R. and Pollefeys, M. (2005). A versatile stereo implementation on commodity graphics hardware. *Journal of Real-Time Imaging*, 11(1):7–18.

Yang, R., Welch, G., and Bishop, G. (2002). Real-time consensus-based scene reconstruction using commodity graphics hardware. In *Pacific Graphics*.

You, S., Hu, J., Neumann, U., and Fox, P. (2003). Urban site modeling from lidar. In *Intl Workshop Computer Graphics and Geometric Modeling (CGGM).*

Zabulis, X. and Daniilidis, K. (2004). Multi-camera reconstruction based on surface normal estimation and best viewpoint selection. In *3DPVT*.

Zabulis, X., Kordelas, G., Mueller, K., and Smolic, A. (2006). Increasing the accuracy of the space-sweeping approach to stereo reconstruction, using spherical backprojection surfaces. In *Int. Conf. on Image Processing*.

Zach, C., Gallup, D., Frahm, J.-M., and Niethammer, M. (2008). Fast global labeling for real-time stereo using multiple plane sweeps. In *VMV*.

Zach, C., Pock, T., and Bischof, H. (2007). A globally optimal algorithm for robust tv-l1 range image integration. In *ICCV*.

Zebedin, L., Bauer, J., Karner, K., and Bischof, H. (2008). Fusion of feature- and area-based information for urban buildings modeling from aerial imagery. In *European Conference on Computer Vision (ECCV).*

Zebedin, L., Klaus, A., Gruber-Geymayer, B., and Karner, K. (2006). Towards 3d map generation from digital aerial images. *ISPRS Journal of Photogrammetry and Remote Sensing*.