A Study of Symmetric and Repetitive Structures in Image-Based Modeling

Jiang Nianjuan

Department of Electronical and Computer Engineering

National University of Singapore

A thesis submitted for the degree of

Doctor of Philosophy

2012 July

Declaration

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis. This thesis has also not been submitted for any degree in any university previously.

Signature:

Date:

Acknowledgements

I would like to offer my sincerest gratitude to all the people who have helped to make this thesis possible.

First of all, I would like to thank Dr. Tan Ping. Most of the work in this thesis was done under close supervision from him. Dr. Tan Ping is a very hard-working and intelligent person. He offered me great help on various problems and difficulties I encountered in my research. I am always inspired by his many bizarre and brave research ideas. It is a great pleasure working with him. Besides research and work, Dr. Tan Ping is also an easy-going and passionate friend in life. The many BBQ outings and conference trips are charitable memories in my PhD life.

I would like to thank Prof. Cheong Loong-Fah. Ever since my undergraduate study in National University of Singapore he has been offering me guidance on computer vision study and research. Prof. Cheong is very knowledgeable and passionate about computer vision research. Under the guidance and supervision of him, I had large freedom on topics I wanted to study and explore. I have received valuable suggestions from him on my thesis writing. I am always grateful to his encouragement for me on pursuing a PhD degree.

In the past five years I have been aided in maintaining the PC hardwares and softwares by Mr. Francis Hoon, a responsible and patient technologist who kept all the lab equipment and facilities in order.

I would like to thank my fellow PhD students and lab colleagues.

They offered help in one way or another on my study and research work. Their cheerful presence made my life as a PhD student so much interesting and enjoyable. Specifically, I would like to thank the following people for assisting in several research experiments. Dr. Gao Zhi helped in the edge detection and segmentation on image patch for my single image modeling project. Mr. Han Shuchu assisted in point cloud alignment and mesh modeling in demonstrating potential applications of symmetry detection project. Mr. Pang Cong helped with early experiments in unambiguous 3D reconstruction project.

I would like to thank the department of Electrical and Computer Engineering for offering me the opportunity and scholarship for my PhD study. Without the financial assistance I would not even start my PhD study.

Beyond research (which sometimes seemed disencouraging and demoralizing) Li Qian had been a companionable housemate for four years. Her cheerful personality always made my home-hour relaxing and fun. I am so happy to have a greate friend like her. Gao Rui has been a great friend ever since I got acquaintance with her. It is a pleasure to have her and her two lovely cats (for not hunting my hamsters and fishes) as my housemates for the past one year.

Finally, I would like to thank my husband, Yunzhen, and my parents for their unconditional understanding and support. It would not have been possible for me to complete my PhD study without their encouragement and love.

Contents

Abstract

Creating photorealistic 3D digital models from street-view imagery has many important applications and involves fundamental vision problems. We investigated the paradox of having similar or repetitive structure in the input image data.

In general, prior knowledge of structure regularity helps with the efficiency and quality of image-based-modeling; however, spurious camera geometries due to appearance ambiguity arising from similar structure can lead to algorithm failure in structure-from-motion, especially for unordered image collections. In this dissertation, we made a detailed survey on 3D reconstruction methodologies and proposed a novel objective function based on 'missing correspondences' to evaluate the optimality of a 3D reconstruction. An efficient algorithm is designed for optimization.

We also investigated the problem on automatic detection of repetitive structures in the recovered scene and proposed a method to jointly analyze images and 3D point clouds to symmetric lattices.

Finally, symmetry is further exploited for a novel camera calibration method and an interactive 3D modeling system working with a single input image.

List of Tables

List of Figures

List of Symbols

$\mathbf{C}$	Camera center	12
$\mathbf{o}$	Camera principal point	12
f	Camera focal length	12
$\mathbf{R}$	Camera rotation matrix	12
$\mathbf{X}$	Homogeneous 4-vector of a 3D point	13
$\mathbf{x}$	Homogeneous 3-vector of a 2D point	13
$\mathbf{P}$	3×4 camera projection matrix	13
$\mathbf{K}$	Camera calibration matrix	13
$\boldsymbol{\omega}$	Image of absolute conic	16
$\boldsymbol{\omega}^*$	Dual image of absolute conic	16
$\boldsymbol{\Lambda}$	Transformation matrix between canonical and real 3D coordinates	18
$\mathbf{W}$	Data matrix of feature correspondences	21
λ	Projective depth	22
s	Scale factor	23
ρ	Penalty for a point being invisible in a view	33

Chapter 1

Introduction

1.1 Background

In the field of computer vision research, we are interested in methods for acquiring, processing, analysing, and understanding images, and in general, high-dimensional data from the real world in order to extract semantic information, e.g. in the forms of decisions. From image analysis, 3D reconstruction, object detection and recognition to scene understanding, it is the ultimate goal of computer vision researchers to duplicate, if not all, but some of the essential capabilities of human visual system by electronically perceiving and understanding the real world environment.

The importance of representation of the scene in computer vision has been debated over the years. In the early years of computer vision research, the reconstructing approach, namely, sense-model-plan-act (SMPA) framework was being criticized as unproductive and impractical (7). The difficulties at that time mainly came from two aspects. First, it was difficult for the computer algorithms to reconstruct or model the scene accurately. Second, the reconstruction process was slow and unresponsive to changes in the environment. With the advance of computer technology and vision algorithms, point features can be detected and matched in sub-pixel accuracy within a fraction of a second (53, 75). A deeper understanding of various numerical problems and successful implementation of mathematical tools such as Bundle Adjustment (90) made 3D reconstruction, once thought as impractical, succeed in various ways. 3D reconstruction can be optimized in speed to achieve simultaneous localization and mapping (SLAM) (40, 61) that builds a 3D representation of the environment in real-time while determining the location with respect to the map in each time instance. Stereo algorithms can be utilized for depth acquisition for autonomous driving systems, e.g. Google driverless car. Offline 3D reconstruction can be optimized for accuracy (83), whereas the obtained 3D clouds can rival with modern laser scanners (78). Other than academic interest, vision technology plays an important role in digital media industry. High quality 3D mesh models are demanded for urban planning, virtual reality (e.g. Google map 3D), digital heritage, movie and game production, etc. These newly emerged digital media dramatically change the way we live and entertain nowadays.

3D reconstruction, or structure-from-motion (SfM), has been studied extensively for more than four decades. Marr and Poggio first proposed the computational theory and algorithm for stereo vision in the late 70s (55), which inspired continuous research effort into stereo algorithms which are the foundation of modern multi-view stereo systems (24). Alternatively, depth information can be recovered from the distribution of apparent velocities of movement of brightness patterns in an image, called optical flow in monocular vision system (e.g. a single

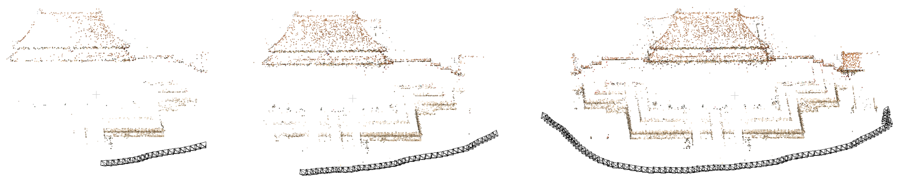


Figure 1.1: Images are added and processed in a sequential manner in incremental 3D reconstruction.

moving camera) (34). With the development of 2D feature trackers such as (29), feature based structure and motion analysis became popular and led to the development of high performance SLAM systems (14, 40, 61). In the 90's, Tomasi and Kanade proposed a factorization framework to solve structure and motion from video sequence under orthographic projection (88). Numerous extension and generalization are proposed in the following decades (6, 65, 71, 85). The advance of view-invariant feature detection and extraction, such as SIFT (53), makes feature correspondences across images with large view change possible. This robust matching capability across different views drew attention from researchers to study camera geometry and SfM for wide baseline stereo and multiple views (30), which are the building blocks for recent well-known 3D reconstruction systems.

However, most well-known 3D reconstruction systems are based on incremental approaches, whereby images are added and processed in a sequential manner Figure 1.1. The image association problem, which is inevitable and error prone in unstructured data collections (e.g. internet images), is often simplified with heuristics in these systems. This simplification often leads to catastrophic failure of the reconstruction in the presence of similar structure and confusing scene appearance, e.g. Figure 1.2. With careful examination, we, as humans, can usually tell the difference if there is sufficient non-ambiguous feature or structure in

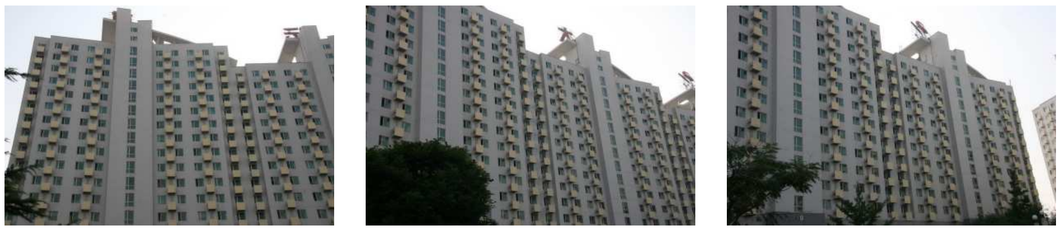


Figure 1.2: Images of ambiguous building structures. It is difficult to tell whether these images describe the same building block or different building blocks with similar appearance.

each image. For instance, different backgrounds and distinctive objects, like the different red sign boards on top of the building in Figure 1.2, suggest observation of different object instances.

Although these similar and repetitive structures cause problem for 3D reconstruction system, they are helpful and much desired for 3D modeling. 3D models are mesh representation of the 3D world, and they are used for all kinds of 3D graphics and rendering applications. In computer graphics, software such as Maya or Google SketchUp are used to create models interactively, images are only used as reference and texture. Internet 3D platforms such as Google Earth and Microsoft Virtual Earth also provide ordinary users with tools to model all kinds of objects on earth. Creating models from scratch is generally time consuming and labour intensive. Images, on the other hand, provide very useful information to assist modeling. 3D models can be directly generated from image silhouettes from multiple calibrated cameras (44), but restricted to small objects with convex surfaces only. Alternatively, we can create 3D models based on the recovered 3D point clouds from 3D reconstruction. However, the recovered 3D point clouds from images are usually sparse and noisy as compared to 3D scanner data, e.g.

Figure 1.3 (d)[1], (e) and (f). Assumptions on the scene geometry properties, such as piece-wise planarity, must be made for efficient modeling (9, 72, 97). The repetitive and symmetric property often exhibited by man-made objects, such as buildings (Figure 1.3 (e) and (f)) provide much stronger constraints for modeling. These properties can be utilized for fast model generation and result in visually appealing high quality 3D models (58, 60). Naturally, automatic detection of these symmetry properties is desirable.

2D symmetry detection from a single image is extensively studied in the past. Methods are developed to detect and categorize rotational symmetry (45, 46, 79), rigid/deformable lattice (32, 50, 51, 52, 54, 67, 69, 95) and bilateral symmetry (12) (Figure 1.3 (a), (b)and (c)). Symmetry can also be directly analyzed from 3D point clouds (4, 11, 57, 70). However, for the purpose of detecting symmetry and regular structure for image-based 3D modeling, all the existing methods face a fundamental difficulty. In the case of 2D symmetry analysis, the presence of perspective distortion makes the image texture asymmetric. Affine invariant features can help with the distortion but fails when there is occlusion, and the repetitive elements appear different in only a single image (Figure 1.3 (f)). 3D symmetry analysis, on the other hand, usually requires laser scanned point clouds which are dense enough for surface normal and curvature computation. Therefore, we study the symmetry detection problem with multiple images and the recovered 3D point clouds obtained in 3D reconstruction. This joint approach also bridges the gap between 2D and 3D symmetry analysis.

When it comes to actual 3D modeling, most existing methods focus on piece-wise planar scenes, since their geometric property is well defined and relatively

[1]Range data is provided by Stanford University Computer Graphics Laboratory.

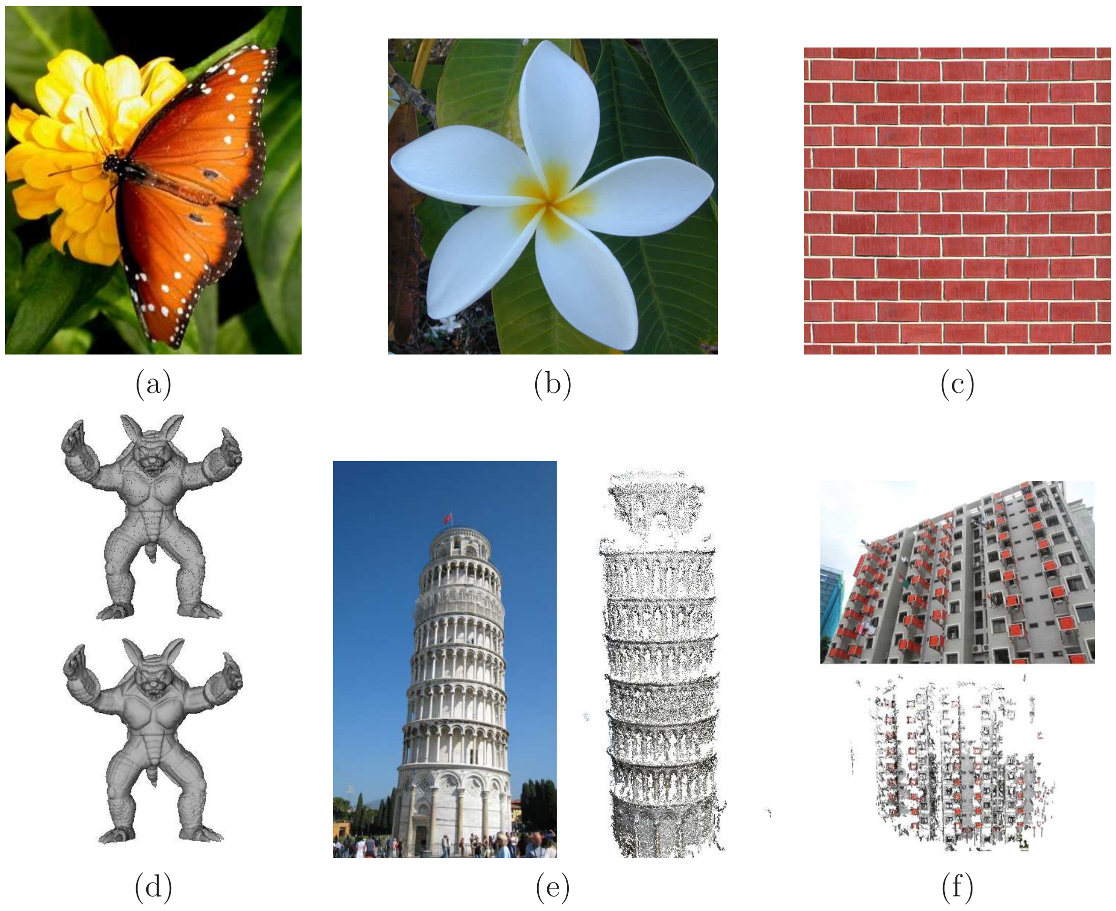


Figure 1.3: (a), (b) and (c) are examples of bilateral symmetry, rotational symmetry and translational symmetry in 2D. (d), (e) and (f) are examples of bilateral symmetry, rotational symmetry and translational symmetry in 3D. The top figure of (d) is the point cloud of laser scanned Armadillo and the bottom figure of (d) is its mesh model. The left figure of (e) is the image of Pisa tower and the right figure of (e) is the point cloud recovered from 3D reconstruction. Same goes for the top figure and the bottom figure of (f) respectively.

easy for automation. Architectures with complex and intricate geometry details and curved surfaces are often modeled interactively and require significant user effort. In our study, we show that symmetry property, e.g. rotational or bilateral symmetry provides very strong geometric constraint on shape and texture, and is sufficient for creating 3D models with complex geometry from as few as a single image. The resulting 3D model can have intricate details and is highly photorealistic.

In summary, the contributions in this thesis consist of the following:

- a detailed survey on 3D reconstruction methodologies
- a novel objective function to evaluate the optimality of a 3D reconstruction and an efficient method for optimization
- a method to jointly analyze images and 3D point clouds to detect repetitive structures and symmetric lattices
- a novel single image calibration method based on 3D symmetry
- an interactive 3D modeling system exploiting 3D symmetry

The study presented in this thesis is also reported in the several publications, (35, 36, 37).

1.2 Thesis overview

The general pipeline of image-based modeling consists of 3D reconstruction, mesh model generation and rendering. 3D reconstruction is the first and most impor-

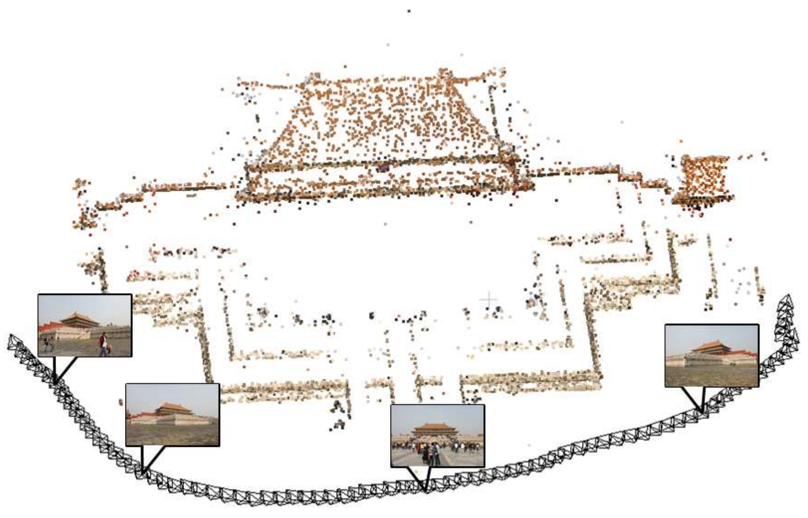


Figure 1.4: 3D reconstruction

tant stage for image-based modeling. In 3D reconstruction, the camera poses and the 3D scene structure (Figure 1.4) are computed.

The most widely used approach for 3D reconstruction from multiple unstructured images is to incrementally integrate new local reconstructions to the global reference frame, i.e. the ordering of the images are required beforehand. Image collections, especially those gathered from the internet, are often unordered. Therefore, the performance of the incremental approach depends on the order the images are associated and integrated into the system. We survey different approaches for 3D reconstruction in Chapter 3 Section 3.1, and discuss their advantages and limitations in handling image association problem. Basic principles for 3D reconstruction are described in Chapter 2. We devote Chapter 3 to a new criteria for evaluating the optimality of a 3D reconstruction, and a novel algorithm for solving the ambiguity in image association and ordering problem. We study the behaviour of the new algorithm both theoretically and empirically.

The point clouds obtained from 3D reconstruction are usually sparse and noisy as compared to 3D scanner data. Geometric constraints such as planarity, orthogonality, parallelism and symmetry are usually used for surface modeling (9, 72, 97). The automatic detection of such geometric constraints is therefore desirable. While the detection of planarity, orthogonality or parallelism can be obtained from geometric analysis and is relatively straightforward, symmetry detection involves higher level of understanding of the scene composition. Symmetry detection is difficult in general, because the input data is never perfect. In 2D symmetry detection, texture analysis could suffer from perspective distortion and occlusion between repetitive objects. Direct analysis on 3D data is impossible without accurate dense point clouds. In Chapter 4, we try to bridge the gap between purely image-based symmetry detection and point-clouds based symmetry detection, and develop an algorithm that works with multiple images with significant perspective foreshortening effect and sparse point clouds.

While most urban architectures consist of planar surfaces and orthogonal edges, there are architectures, especially traditional ones that cannot be modeled well with assumptions of piece-wise planar surfaces, e.g. the ancient Chinese building in Figure 1.1. To make things worse, multiple images may not be always available. Reconstruction and modeling from image(s) of such architectures is still possible if we have proper assumptions. The geometric constraints coming from symmetry alone provide information on the 3D geometry of the object that is under observation (21, 33, 102). We study the geometric constraints of architectures with bilateral and rotation symmetry under perspective camera projection, and exploit such constraints for 3D reconstruction and modeling from a single image. The technical details are described in Chapter 5.

Last but not least, we conclude and discuss limitations of the study presented in this dissertation and issues to be addressed in future research in Chapter 6.

Chapter 2

Principles of 3D Reconstruction

2.1 Camera Calibration

2.1.1 Camera Model

Pinhole camera model A camera is a mapping between the 3D world and a 2D image. The simplest camera model is the basic pinhole camera model, which is also the most commonly used camera model for CCD-like sensors. Figure 2.1 illustrates the central projection of points on to a plane. The center of projection, called the *camera center* or *optical center*, is at the origin of a Euclidean coordinate system and the *image plane* is located at $z = f$. The line from the camera center and perpendicular to the image plane is called the *principal axis* or *principal ray* of the camera. The intersection of principal axis and the image plane is called the *principal point*.

Mathematically, a 3D point can be represented by a homogeneous 4-vector $(X, Y, Z, 1)^T$, and a 2D image point can be represented by a homogeneous 3-

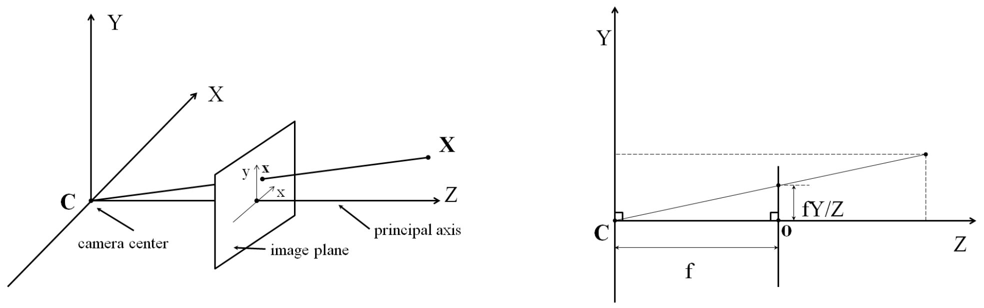


Figure 2.1: Pinhole camera geometry. C is the camera center and $\boldsymbol{o}$ the principal point. The camera here is placed at the coordinate origin. The image plane is placed in front of the camera center and its distance to C is the camera focal length f.

vector $(x, y, 1)^T$. The mapping from a 3D point to a 2D image point by a pinhole camera is expressed as

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \longmapsto \begin{pmatrix} xZ \\ yZ \\ Z \end{pmatrix} = \begin{bmatrix} f & & u & 0 \\ & f & v & 0 \\ & & 1 & 0 \end{bmatrix} [R \;\; -RC] \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}, \tag{2.1}$$

where $x = fX/Z$ and $y = fY/Z$, $\mathbf{R}$ is the rotation matrix that relates the camera coordinate frame and the world coordinate frame as illustrated in Figure 2.2. C corresponds to the world coordinates of the location of the camera center. The *calibration matrix* K is a 3×3 matrix, and

$$\mathbf{K} = \begin{bmatrix} f & & u \\ & f & v \\ & & 1 \end{bmatrix}. \tag{2.2}$$

We denote the homogeneous 4-vector of world point by $\mathbf{X}$, the homogeneous 3-vector of image point by $\mathbf{x}$, and the camera projection matrix by $\mathbf{P}$. Then Equation (2.1) can be rewritten compactly as

$$\mathbf{x} = \mathbf{P}\mathbf{X}, \tag{2.3}$$

where

$$\mathbf{P} = \mathbf{K}[\mathbf{R}\ \mathbf{t}] = \mathbf{K}[\mathbf{R}\ \ -\mathbf{R}\mathbf{C}], \tag{2.4}$$

and we will use this expression throughout the thesis. The parameters contained in $\mathbf{K}$ are called the *intrinsic* camera parameters and the six degrees of freedom contained in $\mathbf{R}$ and $\mathbf{C}$ are called the *extrinsic* camera parameters.

CCD cameras The ideal pinhole camera assumes that the image coordinates are Euclidean coordinates having equal scales in both axial directions. In the case of CCD cameras, it is possible to have non-square pixels. The non-equal scale factors in each direction can be modeled by representing the focal length of the camera in terms of pixel dimensions in the x and y dimensions respectively. Thus, the camera calibration matrix of a CCD camera is

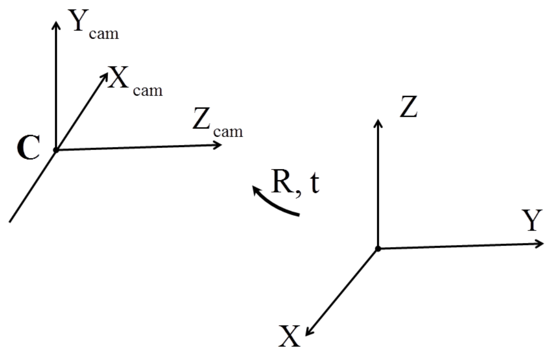


Figure 2.2: The Euclidean transformation between the world and camera coordinate frames.

$$\mathbf{K} = \begin{bmatrix} f_x & & u \\ & f_y & v \\ & & 1 \end{bmatrix}. \tag{2.5}$$

***Skew* parameter** The skew parameter is introduced to take into account the non-perpendicular $x-$ and $y-$ axes of the camera. This is, however, very unlikely to happen for normal CCD cameras. The intrinsic camera matrix with the Skew papameter is written as

$$\mathbf{K} = \begin{bmatrix} f_x & s & u \\ & f_y & v \\ & & 1 \end{bmatrix}. \tag{2.6}$$

Radial distortion In pinhole camera model, the world point, image point and optical center are collinear. For real lenses this assumption will not hold. The deviation observed in normal camera lenses is generally a radial distortion. In practice this error grows as the focal length of the lens decreases. The actual image position after radial distortion (x_d, y_d) is related to the ideal image position $(\tilde{x}, \tilde{y})$ by a distortion factor

$$(x_d, y_d)^T = L(\tilde{r})(\tilde{x}, \tilde{y})^T, \tag{2.7}$$

where $\tilde{r}$ is the radial distance $\sqrt{\tilde{x}^2 + \tilde{y}^2}$. To correct radial distortion, the following equations are used,

$$\begin{aligned} \hat{x} &= x_c + L(r)(x - x_c) \\ \hat{y} &= y_c + L(r)(y - y_c) \end{aligned} . \tag{2.8}$$

The term $L(r)$ is given as a Taylor expension $L(r) = 1 + \kappa_1 r + \kappa_2 r^2 + \kappa_3 r^3 + ...$, and (x_c, y_c) is the center for radial distortion, which is usually taken as the same as the principal point.

2.1.2 Calibration from Homography

Homography is the mapping between different planes. Mathematically, planar point coordinates are transformed by a 3×3 matrix $\mathbf{H}$ as

$$\mathbf{x}' = \mathbf{H}\mathbf{x}. \tag{2.9}$$

The matrix $\mathbf{H}$ can be written as $\mathbf{K}[\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{t}]$, where $\mathbf{r}_1$ and $\mathbf{r}_2$ are the first two columns of $\mathbf{R}$ matrix between the coordinate frame of the plane and the coordinate frame of the camera. A closed form solution of camera intrinsic parameters can be derived based on the orthogonality constraint between $\mathbf{r}_1$ and $\mathbf{r}_2$ (101).

2.1.3 Calibration from Vanishing Points and Lines

The calibration matrix $\mathbf{K}$ relates a image coordinate $\mathbf{x}$ to the direction $\mathbf{d}$ of the ray defined by $\mathbf{x}$ and the camera center, i.e. $\mathbf{d} = \mathbf{K}^{-1}\mathbf{x}$ (30). Hence, the angle between two rays, with direction $\mathbf{d}_1$, $\mathbf{d}_2$ corresponding to image points $\mathbf{x}_1$, $\mathbf{x}_2$ respectively, are given by

$$\begin{aligned} \cos\theta &= \frac{\mathbf{d}_1^T\mathbf{d}_2}{\sqrt{\mathbf{d}_1^T\mathbf{d}_1}\sqrt{\mathbf{d}_2^T\mathbf{d}_2}} \\ &= \frac{\mathbf{x}_1^T(\mathbf{K}^{-T}\mathbf{K}^{-1})\mathbf{x}_2}{\sqrt{\mathbf{x}_1^T(\mathbf{K}^{-T}\mathbf{K}^{-1})\mathbf{x}_1}\sqrt{\mathbf{x}_2^T(\mathbf{K}^{-T}\mathbf{K}^{-1})\mathbf{x}_2}} \end{aligned} \quad . \tag{2.10}$$

The 3×3 matrix $\boldsymbol{\omega} = \mathbf{K}^{-T}\mathbf{K}^{-1}$ is called *the image of the absolute conic* (The absolute conic in metric space is given by identity matrix $I_{3\times 3}$) (30). It follows that if two image points $\mathbf{x}_1$ and $\mathbf{x}_2$ corresponds to orthogonal directions, then

$$\mathbf{x}_1^T\boldsymbol{\omega}\mathbf{x}_2 = 0. \tag{2.11}$$

Under perspective projection, an infinite line is imaged as a line terminating in a *vanishing point*. The vanishing point $\mathbf{v}$ of the normal direction to a plane is related to the plane vanishing line as $\mathbf{l} = \boldsymbol{\omega}\mathbf{v}$. Hence we can also write

$$\mathbf{l}_1^T\boldsymbol{\omega}^*\mathbf{l}_2 = 0, \tag{2.12}$$

where $\boldsymbol{\omega}^* = \boldsymbol{\omega}^{-1}$ is called the dual image of the absolute conic (the DIAC).

In general, five pairs of perpendicular lines are needed to solve for the entries of $\boldsymbol{\omega}$. However, for most cameras we can assume zero-skew and square-aspect ratio. Hence, given entries of $\boldsymbol{\omega}$, where

$$\boldsymbol{\omega} = \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} \\ \omega_{21} & \omega_{22} & \omega_{23} \\ \omega_{31} & \omega_{32} & \omega_{33} \end{bmatrix}, \tag{2.13}$$

we have $\omega_{12} = \omega_{21} = 0$, $\omega_{11} = \omega_{22}$. The remaining entries can be determined from an orthogonal triad of directions. Specifically, the principal point is the orthocentre of the orthogonal triad of vanishing points.

2.1.4 Calibration from Geometric Primitives

Calibration from metric planes The camera calibration matrix $\mathbf{K}$ can be computed from the image of three squares (on planes which are not parallel, but which need not be orthogonal) (30). The algorithm is summarized in the following four steps.

1. For each square compute the homography $\mathbf{H}$ that maps its canonical coordinates of the corner points, $(0,0)^T, (1,0)^T, (0,1)^T, (1,1)^T$, to their imaged points.

2. Compute the imaged circular points (intersection of the plane with the absolute conic) as $\mathbf{H}(1, \pm i, 0)^T$.

3. Fit a conic $\boldsymbol{\omega}$ to the six imaged circular points.

4. Compute $\mathbf{K}$ from $\boldsymbol{\omega}$ using the Cholesky factorization.

Calibration from parallepiped A parallelepiped, as shown in Figure 2.3 is defined by twelve parameters: 3 for orientation, 3 for position, 3 for edge lengths and 3 for angles between parallelepiped edges.

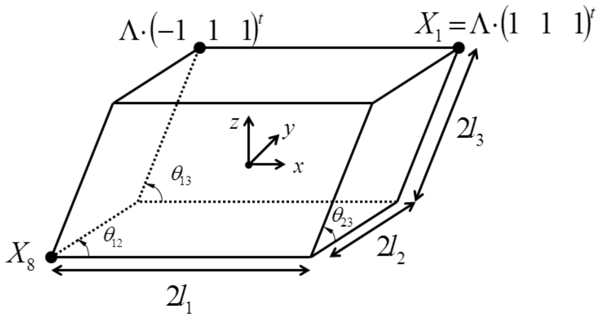


Figure 2.3: Parameterization of a parallelepiped. $2l_i$ are edge lengths, and θ_{ij} are the angles between non-parallel edges.

Given an image of a parallelepiped, the intrinsic characteristics of the camera and those of the parallelepiped give constraints on the parameter sets of both entities(93). Camera projection matrix $\mathbf{P}$ has 11 degrees of freedom and therefore five image points and an image direction are sufficient to determine the projection matrix. The image projection $\mathbf{x}_i$ is related to canonical 3D coordinates by

$$\mathbf{x}_i \sim \mathbf{K}[\mathbf{R}\ \mathbf{t}]\mathbf{\Lambda}. \tag{2.14}$$

The matrix $\mathbf{\Lambda}$ can be written as

$$\mathbf{\Lambda} = \begin{pmatrix} l_1 & l_2 c_{12} & l_3 c_{13} & 0 \\ 0 & l_2 s_{12} & l_3 \frac{c_{23} - c_{13} c_{12}}{s_{12}} & 0 \\ 0 & 0 & l_3 \sqrt{\frac{s_{12}^2 - c_{13}^2 s_{12}^2 - (c_{23} - c_{13} c_{12})^2}{s_{12}^2}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{2.15}$$

with $c_{ij} = \cos\theta_{ij}, s_{ij} = \sin\theta_{ij}, \theta_{ij} \in [0\pi], l_i > 0$. Let $\mathbf{P}_0$ and $\mathbf{\Lambda}_0$ represent the matrices of the first three lines and columns of $\mathbf{P}$ and $\mathbf{\Lambda}$, then the intrinsic camera parameters and the parallelepiped's parameters are related by

$$\mathbf{P}_0^T \mathbf{K}^{-T} \mathbf{K}^{-1} \mathbf{P}_0 = \mathbf{P}_0^T \boldsymbol{\omega} \mathbf{P}_0 \sim \boldsymbol{\Lambda}_0^T \boldsymbol{\Lambda}_0. \tag{2.16}$$

Prior knowledge on θ_{ij}, l_{ij} or camera intrinsic parameters give rise to linear or quadratic constraints on the rest of the parameters. As reported in (93), the focal length estimation is not sensitive to the assumption of location of the principal point, but degrades quadratically with aspect ratio error.

2.2 3D Reconstruction

2.2.1 Two-View 3D Reconstruction

Camera geometry between two images from different viewpoints is also called *epipolar geometry*, which only depends on the cameras' intrinsic parameters and relative pose. As shown in Figure 2.4, corresponding image points $\mathbf{x}$ and $\mathbf{x}'$ in view i and i' are related to each other via the epipolar plane that passes through 3D point $\mathbf{X}$, camera center $\mathbf{C}$ and $\mathbf{C}'$. They satisfy the equation given as follows,

$$\mathbf{x}'^T \mathbf{F} \mathbf{x} \;=\; 0. \tag{2.17}$$

The 3×3 matrix $\mathbf{F}$ can be further decomposed as $\mathbf{F} = \mathbf{K}'^{-T} \mathbf{E} \mathbf{K}^{-1}$, where $\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}$ captures the rigid transformation between the two cameras and is of rank 2 ($[\mathbf{t}]_\times$ is the skew-symmetric matrix of vector $\mathbf{t}$).

The fundamental matrix $\mathbf{F}$ can be computed from feature correspondences alone by solving a linear system given eight pairs of feature correspondences according to Equation (2.17). The computation for fundamental matrix is degenerate when the scene points lie on a ruled quadric or on a plane (30), the latter is

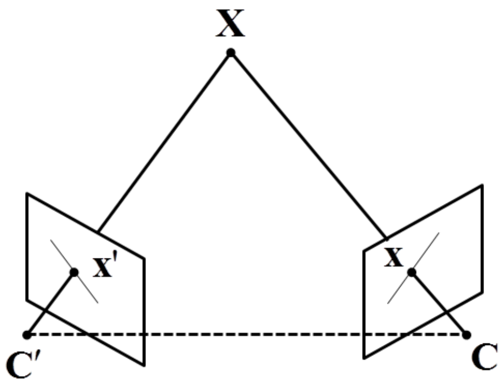


Figure 2.4: Epipolar geometry.

the common case for reconstructing architectural objects. If the camera intrinsic parameters are known, Equation (2.17) is reduced to

$$\hat{\mathbf{x}}'^T \mathbf{E} \hat{\mathbf{x}} = 0, \tag{2.18}$$

where $\hat{\mathbf{x}}$ is the calibrated image point given by $\mathbf{K}^{-1}\mathbf{x}$. $\mathbf{E}$ can be computed robustly from five image correspondences by exploiting the rank 2 property and orthogonality of the rotation matrix $\mathbf{R}$ (62). This five-point algorithm generally does not suffer from planar degeneracy and is widely used in most well-known 3D reconstruction systems. 3D points are computed from feature correspondences and camera parameters via triangulation.

2.2.2 Multi-View 3D Reconstruction

When there are more images, the reconstruction methods can be classified into three categories in general, namely,

- solve for global camera poses and 3D structure at the same time from matrix factorization of all feature correspondences.
- solve for all camera rotations followed by all camera translations and 3D structure.
- solve for camera poses and 3D structure incrementally.

The Factorization algorithm

Given feature tracks through the image sequence, one can stack all the feature correspondences into a big matrix in the following form,

$$\mathbf{W} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ y_{11} & y_{12} & \cdots & y_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ y_{21} & y_{22} & \cdots & y_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \\ y_{n1} & y_{n2} & \cdots & y_{nm} \end{bmatrix}. \tag{2.19}$$

Under orthographic projection[1] the data matrix $\mathbf{W}$ can be directly decomposed as $\mathbf{W} = \hat{\mathbf{P}}\hat{\mathbf{S}}$, where $\hat{\mathbf{P}}$ is a $2n \times 2$ matrix and $\hat{\mathbf{S}}$ is a $2 \times m$ matrix, this is called the Tomasi-Kanade factorization (88). Metric reconstruction can be recovered from $\hat{\mathbf{P}}$ and $\hat{\mathbf{S}}$ by using the orthogonal property of the $x-$ and $y-$ axis of the camera. Under perspective projection (the pinhole camera model), the data matrix needs to have the following form to perform factorization (89),

[1]Refer to (30) for detailed introduction to different camera projection models.

$$\mathbf{W} = \begin{bmatrix} \lambda_{11}x_{11} & \lambda_{12}x_{12} & \cdots & \lambda_{1m}x_{1m} \\ \lambda_{11}y_{11} & \lambda_{12}y_{12} & \cdots & \lambda_{1m}y_{1m} \\ \lambda_{11} & \lambda_{12} & \cdots & \lambda_{1n} \\ \lambda_{21}x_{21} & \lambda_{22}x_{22} & \cdots & \lambda_{2m}x_{2m} \\ \lambda_{21}y_{21} & \lambda_{22}y_{22} & \cdots & \lambda_{2m}y_{2m} \\ \lambda_{21} & \lambda_{22} & \cdots & \lambda_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{n1}x_{n1} & \lambda_{n}x_{n2} & \cdots & \lambda_{nm}x_{nm} \\ \lambda_{n1}y_{n1} & \lambda_{n2}y_{n2} & \cdots & \lambda_{nm}y_{nm} \\ \lambda_{n1} & \lambda_{n} & \cdots & \lambda_{nm} \end{bmatrix}, \tag{2.20}$$

where λ_{ij} is called the projective depth of the image point $\mathbf{x}_{ij}$ in view i. It is related to the true point depth by an arbitrary scale. The $3n \times m$ matrix $\mathbf{W}$ in Equation (2.20) can be decomposed as $\mathbf{W} = \mathbf{PH}^{-1}\mathbf{HS}$, where $\mathbf{P}$ consists of the camera projection matrices for m views in metric frame, $\mathbf{S}$ consists of homogeneous coordinates of the recovered 3D points and $\mathbf{H}$ is an arbitrary 4×4 projective transformation. The projective depths are usually unknown and need to be recovered together with the camera matrices and 3D points, this is usually done in an iterative fashion by alternating between projective depths estimation and structure-and-motion estimation (65, 85). There are many variations of factorization method that handle missing and outlier entries in $\mathbf{W}$ (8, 39, 64).

Two-stage solution

Given pair-wise epipolar geometries, the structure-from-motion problem can be solved in a two-stage fashion. In the first stage, relative rotations between pairwise cameras are used as constraints to solve for absolute rotation [2] of each individual camera. Let $\mathbf{R}_i$ and $\mathbf{R}_j$ denote the absolute rotation of camera i and j, R_{ij} denote the relative rotation between them, we have

$$\mathbf{R}_j = \mathbf{R}_i\mathbf{R}_{ij}. \tag{2.21}$$

The absolute positions of camera i and j are related to the relative translation vector $\mathbf{t}_{ij}$ by

$$\mathbf{R}_j(\mathbf{C}_i - \mathbf{C}_j) \;=\; s_{ij}\mathbf{t}_{ij}, \tag{2.22}$$

where $\mathbf{t}_{ij}$ is a unit vector, and s_{ij} is an arbitrary scale factor.

In the second stage, Govindu (26) proposed to use a relaxed version of Equation (2.22) that eliminates s_{ij} by using cross product of the two sides in Equation (2.22) and obtain

$$[\mathbf{t}_{ij}]_\times\mathbf{R}_j(\mathbf{C}_i - \mathbf{C}_j) \;=\; \mathbf{0}. \tag{2.23}$$

One can thus solve for absolute camera translations using the heading directions between view pairs without solving for the individual scale explicitly. However, the relaxed version of Equation (2.22) may suffer from degeneracy in the case where the camera is moving along a single direction, e.g. camera mounted car moving along a straight street.

[2]Note that this is not the absolute physical orientation of the camera. We can only recover metric structure up to a rigid transformation and scaling of the world coordinate frame for any structure-from-motion algorithm. The same explanation goes for camera translation.

Kahl (38) reformulated the translation estimation problem as a quasi convex problem of the form $\frac{f_1(x)^2+f_2(x)^2}{\lambda(x)^2}$, where $f_1(x)$, $f_2(x)$ and $\lambda(x)^2$ are affine functions with coefficients determined from the entries in absolute camera rotation matrices $\mathbf{R}_i$s' and vectors of image feature points. L_∞ norm is used as error function for optimization. L_∞ norm is however, very sensitive to gross errors in the data. The translation estimation can be done more robustly and efficiently by careful outlier analysis and feature reduction on the initial correspondences as described in (56).

Scale and translation constraints can be established within view triplets based on pair-wise reconstructions. Sinha et al. (80) leveraged on these constraints on view triplets sharing common 3D points to derive a linear solution for global translation estimation. In fact, each pair-wise camera poses differ from the global camera poses by a scale s_{ij} and translation $\mathbf{t}_{ij}$ after global rotation alignment in the first stage. Given pair-wise reconstruction between image pair ij and jk, the following equations can be easily derived,

$$\begin{aligned} s_{jk} - s_{ij}^{jk} s_{ij} &= 0 \\ s_{jk} t_{jk} &= s_{ij}^{jk} \mathbf{t}_{ij} + \mathbf{t}_{ij}^{jk}, \end{aligned} \tag{2.24}$$

where s_{ij}^{jk} and $\mathbf{t}_{ij}^{jk}$ are the scale and translation between pair-wise reconstruction from image pair ij and jk respectively. Sinha et al. (80) solved a weighted equation system of Equation (2.24) from the largest connected component of connected image pairs. This formulation avoid solving 3D points in translation estimation and can be solved much more efficiently as compared to the approach described in (38).

Feature mismatches and erroneous pair-wise epipolar constraints will cause failure of most batch solutions that are sensitive to gross error. Crandall et al. (13) used GPS data as additional constraints on the camera poses and proposed a hybrid global optimization framework that solves a discrete labeling problem and a nonlinear least square optimization.

Incremental solution

Batch solutions usually require prior knowledge of structured image data, i.e. known camera intrinsic settings and sequential information. Image data collected from the internet, which is abundant and ideal for city reconstruction applications, are usually unstructured. Very little prior knowledge of the camera settings is known and the photos taken from different sources are not in any particular order. Therefore, most SfM systems for unordered photo collections are incremental, e.g. (1, 41, 48, 82, 83). An incremental solution usually starts with a small reconstruction, then grows a few images at a time, triangulate new points, and does one or more rounds of nonlinear least squares optimization (known as Bundle Adjustment (90)) to minimize the reprojection error. This process is repeated until no more images can be added.

Chapter 3

Unambiguous Multi-view 3D Reconstruction

3.1 SfM from Unordered Image Collection

3.1.1 Overview

Structure-from-motion (SfM) algorithms estimate both camera poses and 3D structures of a scene from images. SfM with unordered image sets such as internet images is a challenging task. One needs to first determine how the images are matched against one another. This image matching problem can be thought of as a graph estimation problem where we are given a set of vertices corresponding to the images and we need to discover the set of edges connecting them. Usually, an edge connects a pair of images if and only if they are looking at the same part of the scene and have a sufficient number of feature correspondences where a valid epipolar geometry can be computed. This graph can be called a 'match graph' or 'connectivity graph'. Invalid connections between images arising from incorrectly computed epipolar geometries will cause catastrophic failure for both incremental solutions and batch solutions. Therefore, it is critical to identify and remove them.

Erroneous epipolar geometries could arise from: 1) degenerate configuration in relative pose computation, 2) matching failure due to feature descriptors, or 3) duplicate structures in the scene. In the first two cases, the incorrect epipolar geometries are often independent and inconsistent from each other, and can be detected by local geometric consistency verification such as trifocal geometry verification a mong image triplets (31). When the percentage of the incorrect epipolar geometries is small, Martinec and Pajdla (56) identified them by checking the residual in global rotation and translation registration. Alternatively, loop consistency analysis of camera rotation (99), (18) can be applied. However, when there are duplicate structures in the scene, they could generate a large set of incorrect epipolar geometries that are consistent with each other, which makes the aforementioned motion consistency check fail. Such an example is provided in Figure 3.1, where multiple images are captured around a cup. In the top of Figure 3.1, we connect two cameras by a line segment, if an epipolar geometry can be computed between them. (Note that we do not exhaustively draw all these line segments to make the illustration clear.) Images of the two different sides of the cup can match and generate many incorrect epipolar geometries. One of such image pair is shown in the red rectangle on the left. The green rectangle on the right shows a correctly matched image pair. The incorrect image pairs overwhelm the correct ones in number, and as a result, previous methods such as (83, 99) will generate incorrect results as shown in the bottom row, where all the

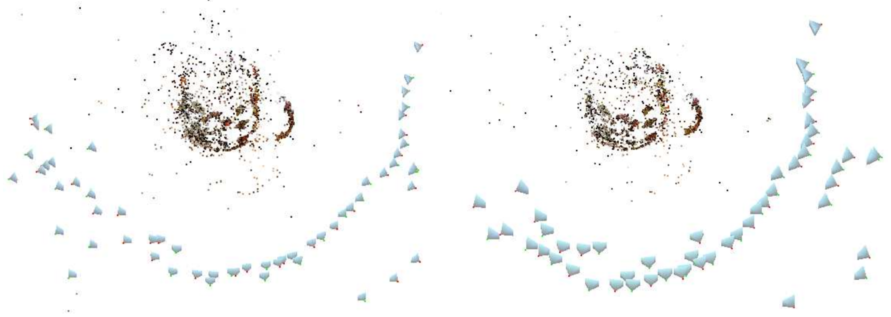


Figure 3.1: The middle of first row shows the true configuration in which multiple images are captured around a cup. We highlight a pair of correct and incorrect image matches in the green and red rectangles respectively. The second row are the reconstructions obtained using (83) and (99) respectively. All cameras are incorrectly reconstructed on one side of the cup.

cameras are reconstructed on one side of the cup.

This problem was solved in (74) by using image timestamps and 'missing correspondences' in local image neighbourhood. However, image timestamps can only be applied to sequentially captured data. Missing correspondences analysis in image triplets was first introduced in (98) to locally identify incorrect image pairs from a third image. However, as the authors acknowledged in their paper (74, 98), incorrect pairs may also pass this local verification.

In this study, we argue that the 'missing correspondences' suffices to solve the visual ambiguity when analyzed in a more holistic fashion. Instead of analyzing locally within a triplet as in (98), we propose a novel objective function that evaluates the overall quality of a 3D reconstruction by using the missing correspondences. We first demonstrate the global minimum of this objective function is associated with the correct 3D reconstruction, and then show an efficient method to optimize this objective function.

Given a set of unordered images, we first construct a match graph based on existence of pairwise epipolar geometries, where each vertex is a camera and two cameras are connected if a valid epipolar geometry with sufficient number of inlier point correspondences can be computed between them. Each edge is weighted by the reciprocal of the number of correspondences between its image pairs. A spanning tree on the match graph determines a 3D reconstruction. Hence, we search in the space formed by all spanning trees. We start from the minimum spanning tree, and iteratively identify possible problematic edges and replace them by favorable ones to minimize our objective function. The algorithm stops when no spanning tree with better score can be found. In our algorithm, each iteration always decreases the non-negative objective function; thus convergence is guaranteed. The convergence is also typically fast, because the number of iteration required is bounded by the number of different 3D reconstructions arising from those ambiguous epipolar geometries, which is often not too large in real data.

Our main contributions in this study are twofold. First, we design an objective function that correctly describes the optimality of a reconstruction. Second, we design an efficient optimization of this objective function, and demonstrate the superiority of our approach compared to the state-of-the-art.

3.1.2 Related Works

Detection of incorrect epipolar geometries is crucial for SfM algorithms. Existing methods that detects invalid connectivity between images used can be roughly categorized into three types or a combination of these types:

- local heuristics
- geometric consistency verification

- 'missing correspondence' cue

Local heuristics are used commonly in typical incremental SfM systems such as 'Bundler' (83) to determine the connectivity and ordering of images. Schaffalitzky et al. (77) combined image invariants/covariants and geometric relations to organize unordered image sets of multiple non-overlapping scenes for image browsing in 3D. Martinec and Pajdla (56) and Sinha et al. (80) both addressed this problem implicitly in a global registration framework. The former iteratively discarded the image pair with the highest residual, while the latter weighted different epipolar constraints using the number of triplet-consistent points. Li et al. (48) used maximum spanning tree on the match graph to determine the order of image registration, where match graph edges were weighted by the number of correspondences. All these methods only work when the percentage of incorrectly matched image pairs is small.

To handle more incorrect image pairs, both Havlena et al. (31) and Klopschitz et al. (41) performed reconstruction with submodels obtained from view triplets. Zach et al. (99) inferred the validity of epipolar geometries by evaluating loop consistency in the match graph. Govindu (27) adopted a sampling approach in the spirit of RANSAC to sample spanning trees and select the largest set of self-consistent epipolar geometries. All these methods implicitly assume that the erroneous epipolar geometries are statistically independent and inconsistent, and are relatively few in number as compared to the correct ones. Thus, these methods fail on data with a large number of incorrect epipolar geometries arising from duplicate scene structures. Recent work (13) incorporated GPS data as additional constraint to initialize the SfM problem globally. Epipolar geometries inconsistent with the global motion were identified as outliers and removed from subsequent computation.

Zach et al. (98) first proposed to analyze 'missing correspondences' among image triplets to identify wrong image matches. Roberts et al. (74) incorporated this cue to assist an Expectation-Maximization based estimation of the correctness of each image pair. However, both of them only analyze missing correspondences locally, and cannot identify all incorrect epipolar geometries. While Roberts et al. (74) resorted to image timestamps to solve the problem, their approach is not applicable to non-sequentially captured images.

Data association problem is also extensively studied in simultaneous localization and mapping (SLAM) (3, 25, 73). SLAM algorithms must detect reoccurrence of previously observed scenes, and decide whether it is due to loop closure or duplicate scene structures. Due to the sequential nature of SLAM images, this decision is much easier to make than the case for unordered images.

3.2 Quantitative Reconstruction Evaluation

3.2.1 Objective function

Intuitively, in a correct reconstruction, a 3D point should have similar appearance in images where it is visible. An approximate surface normal can be computed for each 3D point using patch-based stereo (24). We define a SIFT descriptor(53) for a reconstructed 3D point as the SIFT descriptor of the image feature point in its most front parallel image (with respect to the normal associated with the point). If a 3D point is visible in an image, its SIFT descriptor should match with the SIFT descriptor evaluated at its image projection. Therefore, the validity score of a 3D reconstruction can be defined as

$$E_R = \frac{1}{m}\sum_{p=1}^{m}\hat{P}_{missing}(p) = \frac{1}{m}\sum_{p=1}^{m}\frac{1}{n}\sum_{i=1}^{n}P_{missing}(p,i). \tag{3.1}$$

where m is the total number of reconstructed 3D points, and n is the total number of images. $\hat{P}_{missing}(p)$ is the average of $P_{missing}(p,i)$ over all images, and $P_{missing}(p,i)$ is the probability that the SIFT descriptor of p does not match with that of its image projection $proj_{p,i}$ in view i. We define

$$P_{missing}(p,i) = \begin{cases} 0 & \text{if } p \text{ finds matched image feature in view } i \\ 1 & \text{if } p \text{ finds no matched image feature in view } i \end{cases}. \tag{3.2}$$

In practice, we set the searching window for candidate matching feature points to 100×100 centered around $proj_{p,i}$ (with image resolution about 1200×800)[1]. Visibility issue needs to be resolved before we project point p into view i for appearance similarity test, and we will discuss this in more detail in Section 3.2.2. To account for matching failures and mismatches, we also penalize p being invisible in the image i by setting $P_{missing}(p,i) = \rho$ (we use $\rho = 0.05$ in all our experiments). Hence, the complete definition of $P_{missing}(p,i)$ is given as

[1]We threshold on the angle between two SIFT descriptors to decide if there is a match. Since the matching ability of SIFT descriptor decreases quickly as the view change gets large, we use two thresholds: 50° if the view change is less than 45° (with respect to the reference view of the 3D point), and 60° if the view change is between 45° to 60°.

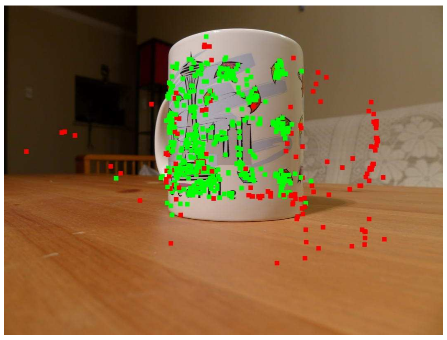
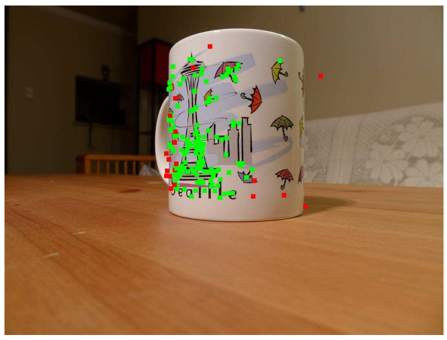
(a)  (b) 

Figure 3.2: Missing correspondence analysis, where 'missing points' are marked in red. Shown is a view of a cup that forms a triplet with the cameras pair (Cam_l, Cam_r) highlighted in red in Figure 3.1. (a) In our formulation, we check all the reconstructed 3D points in all images. A large amount of 'missing correspondences' can be identified for the 3D reconstruction corresponding to a spanning tree containing (Cam_l, Cam_r) as the only erroneous pair; (b) Local triplet analysis according to (98) fails to identify the incorrect image pair.

$$P_{missing}(p,i) = \begin{cases} 0 & \text{if } p \text{ finds matched image feature in view } i \\ 1 & \text{if } p \text{ finds no matched image feature in view } i \\ \rho & \text{if } p \text{ is invisible in view } i \end{cases} . \tag{3.3}$$

For easy reference, we refer $proj_{p,i}$ as a 'consistent'/'inconsistent' point respectively when a match can/cannot be found. An example is illustrated in Figure 3.2 (a), where consistent and inconsistent points are marked in green and red respectively.

E_R evaluates the average likelihood that a reconstructed 3D point is missing in the images. Ideally, in a correct reconstruction, this probability should be zero. In real data, it is often a small positive value because of the imperfect feature registration. In comparison, incorrect 3D reconstruction with erroneous image matches will result in a large positive E_R. Thus, intuitively, the global minimum of E_R should correspond to a correct 3D reconstruction.

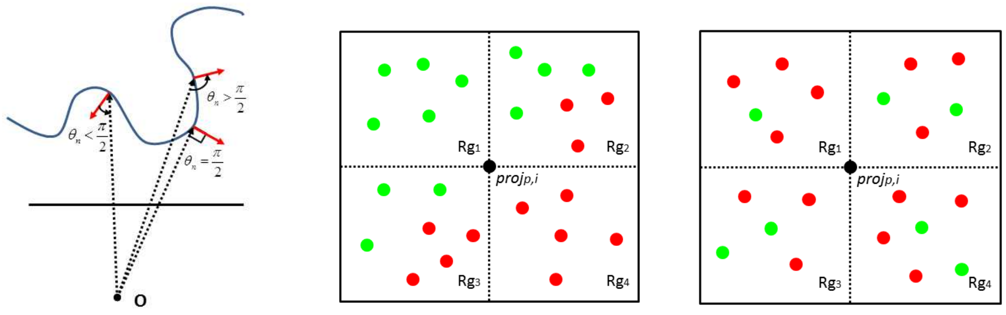
The definition of E_R is similar to the 'missing correspondences' in (98). The key difference is that we evaluate E_R comprehensively over all reconstructed points and all images. In comparison, Zach et al. (98) evaluated 'missing correspondences' triplet by triplet to identify incorrect image pairs locally. Local triplet verification cannot identify some incorrect image pairs. For example, the image in Figure 3.2 (b) forms a triplet with the incorrectly matched image pairs in the red rectangle in Figure 3.1. These three cameras are marked by red in Figure 3.1. However, there is little 'missing correspondences' in Figure 3.2 (b). Hence, this triplet will be considered as correct in (98). In comparison, we evaluate E_R on the complete 3D reconstruction (resulting from a spanning tree with only one erroneous edge as in the triplet). Many inconsistent points can be identified in Figure 3.2 (a).

3.2.2 Visibility test

Before computing the probability of $proj_{p,i}$ being inconsistent, we need to know the visibility of point p in view i. A point p is invisible in view i if

- p is out of the field of view of camera i, or
- p is on a surface face away from camera i, or
- p is occluded.

Recall that we can compute patch orientation of point p from image pairs, so we can use the difference between point patch orientation and its line-of-sight to determine whether point p is face away from camera i. This is illustrated in


$\theta_n < \frac{\pi}{2}$
$\theta_n > \frac{\pi}{2}$
$\theta_n = \frac{\pi}{2}$
O
Rg1
Rg2
Rg3
Rg4
projp,i

(a) (b) (c)

Figure 3.3: Visibility test. (a)Point is considered as invisible if the angle between the surface normal and the line-of-sight is greater than 90°. (b)Point p is considered as visible in view i. (c)Point p is considered as invisible in view i.

Figure 3.3 (a). In particular, we denote the angle between the surface orientation of point p and the its line-of-sight in view i as θ_n, and p is considered as invisible in view i if θ_n is greater than 90°.

For occlusion detection, we use a simple statistic to determine point p's visibility. We compare the depth of all point projections in a 100×100 window centered at $proj_{p,i}$ to that of $proj_{p,i}$. In Figure 3.3 (b), we mark point projections with smaller depths (the different between depths should be larger than $frac_1 20$ of the depth of point p, assuming 1000 pixel focal length, 1 pixel image noise and minimum 2 degrees of triangulation angle) as compared to that of $proj_{p,i}$ (points in front of p) as red, and the rest (points behind p) as green. Point p is considered as occluded in view i if all four regions are populated by points in front of p.

3.2.3 Objective Function Validation

We first validate our objective function in Equation (3.1) with a number of real data to demonstrate that its global minimum is often associated with the true 3D reconstruction. For each of the examples in Figure 3.5, Figure 3.6 and Figure 3.7, we obtain up to 100 different 3D reconstructions and evaluate the objective function Equation (3.1) on these results. To obtain these different 3D reconstructions, we randomly sample spanning trees from the match graph. Each spanning tree gives a 3D reconstruction of the scene. We further require these 3D reconstructions to be different from each other (see more details in Section 3.3). Besides these randomly sampled spanning trees, we also manually specify a spanning tree with only correct epipolar geometries to obtain the 'ground truth' result. We then evaluate the objective function for each 3D reconstruction. We sort these results in ascending order and plot them in Figure 3.4 (a). We mark the position of the ground truth reconstruction by a square. Clearly, among these 100 different 3D reconstructions, the 'ground truth' result always leads to the smallest value of the objective function. This gives a strong indication that the global optimal of Equation (3.1) is associated with the true configuration. It suggests that we can obtain the correct solution by searching the space of all spanning trees and choosing the one with minimum cost.

3.3 Efficient Optimization

Given the objective function, we want to minimize it to seek a correct 3D reconstruction. Starting from epipolar geometries computed between image pairs[2], we perform triplet geometry consistency verification as in (74). We only keep epipolar geometries that are supported by at least one view triplet. For each image pair with valid epipolar geometry computed, we further reconstruct 3D

[2]We apply RANSAC(19) and the five-point algorithm(62) for computation. We consider an epipolar geometry exists if at least 30 points with reprojection error less than 4 pixels can be reconstructed.

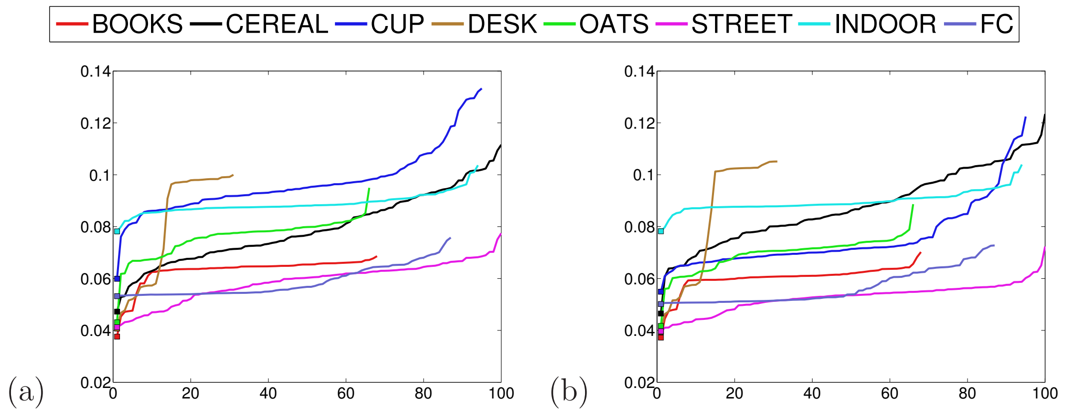


Figure 3.4: Objective function evaluation. We check up to 100 different 3D reconstructions for each example in Figure 3.5, Figure **??** and Figure **??**, and plot the objective function values of these reconstructions in ascending order. The value for 'ground truth' is marked by a square. (a) and (b) show the plotting with Equation (3.1) and Equation (3.9) respectively.

points with rough orientations from their feature matches (24). The 3D points are represented by the depth of feature points in both images.

We define a match graph, where each camera is a vertex and two cameras are connected if an valid epipolar geometry can be computed between them. We assume the graph has only one connected component, though we can process component by component otherwise. Each edge of the match graph is then associated with a weight $\frac{1}{m_{ij}}$, m_{ij} is the number of reconstructed 3D points between i and j. We look for a spanning tree of the match graph to minimize our objective function. We choose the minimum spanning tree to initialize this search, and compute the 3D reconstruction from it according to (83). Bundle adjustment is performed to refine the relative camera poses. After this refinement, the initial objective function is evaluated.

We greedily search for a better spanning tree from a given starting point. We design a strategy to ensure that the whole process is efficient. First, we notice that different spanning trees could lead to the same 3D reconstruction. To avoid repetitively evaluating equivalent trees, we cache visited 3D reconstructions and only search trees that lead to different 3D reconstructions. Second, at each step of the iterative search, we replace only one edge of the spanning tree to move to a new tree, such that the two successive trees are similar and we can reuse the computation in 3D reconstruction. Third, we further provide an alternative definition of the objective function to facilitate its evaluation. In the following, we will introduce these methods in turn.

3.3.1 3D Reconstruction Caching

Given a spanning tree, we can classify all the epipolar geometries as consistent or inconsistent with it. We record all consistent epipolar geometries for each visited spanning tree using a binary array. Given a new tree, if all the epipolar geometries associated with its edges are consistent with another tree that has been previously visited, we consider this new tree as redundant and skip it.

In the following, we explain how to decide if an epipolar geometry is consistent or inconsistent with a given spanning tree. This is essentially similar to the loop consistency verification in (99). Given a spanning tree, the relative motion between any two cameras can be derived by chaining the relative motions from pairwise epipolar geometries along the tree path. Let $L = i, l_1, l_2, l_3, \cdots, l_k, j$ represent the vertices on the tree path that connects view i and view j, then we have the relationship between relative rotations and absolute rotations,

$$\begin{aligned}
\mathbf{R}_{l_1} &= \mathbf{R}_i \mathbf{R}_{il_1} \\
\mathbf{R}_{l_2} &= \mathbf{R}_{l_1} \mathbf{R}_{l_1 l_2} \\
&\cdots \\
\mathbf{R}_j &= \mathbf{R}_{l_k} \mathbf{R}_{l_k j},
\end{aligned} \tag{3.4}$$

and the relationship between relative camera translation and absolute camera poses,

$$\begin{aligned}
\mathbf{C}_{l_1} &= \mathbf{R}_{il_1}^T (\mathbf{C}_i - s_{il_1} \mathbf{t}_{il_1}) \\
\mathbf{C}_{l_2} &= \mathbf{R}_{l_1 l_2}^T (\mathbf{C}_{l_1} - s_{l_1 l_2} \mathbf{t}_{l_1 l_2}) \\
&\cdots \\
\mathbf{C}_j &= \mathbf{R}_{l_k j}^T (\mathbf{C}_{l_k} - s_{l_k j} \mathbf{t}_{l_k j}).
\end{aligned} \tag{3.5}$$

Note that $s_{il_1}, s_{l_1 l_2}, \cdots, s_{l_k j}$ are the baseline lengths between camera pairs in the registered global camera motion and they cannot be determined from individual epipolar geometry. We follow (74) to determine baseline lengthes. Specifically, we form a tree of triplets according to the spanning tree (with each node representing a triplet and each edge being an edge from the spanning tree and shared by the two triplets associated with its two nodes) and traverse this tree of triplets to decide the baselines of child triplets according to that of their parent. Furthermore, the baseline between each camera pair is computed only once according to the first visited triplet containing that camera pair.

On the other hand, we can also compute the relative motion between view i

and view j from their own epipolar geometry, i.e.,

$$\begin{aligned} \mathbf{R}_j &= \mathbf{R}_i\mathbf{R}_{ij} \\ \mathbf{C}_j &= \mathbf{R}_{ij}^T(\mathbf{C}_i - s_{ij}\mathbf{t}_{ij}). \end{aligned} \tag{3.6}$$

Hence, we have two relative motions between camera i, j, namely, $(\hat{\mathbf{R}}_{ij}, \hat{\mathbf{t}}_{ij}, \hat{s}_{ij})$ from chaining the epipolar geometries along the tree path, and $(\mathbf{R}_{ij}, \mathbf{t}_{ij}, s_{ij})$ from the epipolar geometry between view i and view j and the baseline length from the first rescaled triplet containing view i, j.

We can then determine an epipolar geometry as consistent or inconsistent according to the agreement between these two relative motions. We compute the probability of an edge being inconsistent as,

$$Prob(\mathbf{e}_{ij} \in S^c) = e^{-\beta \mathbf{V}_{ij}^T \Sigma^{-1} \mathbf{V}_{ij}}, \tag{3.7}$$

where S^c indicate the set of inconsistent edges, β is a constant (we set $\beta = 0.1$), Σ is the covariance matrix, and $\mathbf{V}_{ij} = 1/(\max(L/L_0, 1)) \left(\widehat{\mathbf{r}}_{ij}, \widehat{\mathbf{t}}_{ij}, \widehat{s}_{ij}\right)^T$ is the motion discrepancy vector between camera i, j. $\widehat{\mathbf{r}}_{ij}$ is the orientation difference of the two relative rotations (calculated as the average angular difference between the corresponding rows of the two relative rotation matrices); $\widehat{\mathbf{t}}_{ij}$ is the orientation difference between the two relative translations, and $\widehat{s}_{ij}$ is the baseline length difference normalized by the average baseline length of immediate adjacent cameras on the spanning tree. The covariance matrix Σ is computed from motion discrepancy vectors $\mathbf{V}$ obtained from geometrically consistent triplets. To account for drifting effects, we further divide $\widehat{\mathbf{r}}_{ij}$, $\widehat{\mathbf{t}}_{ij}$ and $\widehat{s}_{ij}$ by L/L_0, when $L > L_0$. Here L is the distance between the two cameras i and j along the spanning tree, L_0

is chosen to be 6 (same as the maximum loop length in (99)). All edges with $Prob\left(\mathbf{e}_{ij} \in S^c\right) > 0.5$ are considered inconsistent and assigned to S^c.

3.3.2 Incremental Spanning Tree Search

At each step we break one edge $\mathbf{e}_{off}$ from the existing spanning tree, and add another edge $\mathbf{e}_{on}$ to connect the two subtrees T_l and T_r generated by removing $\mathbf{e}_{off}$. The relative camera poses within T_l and T_r are unchanged during this process. Hence, we can reuse the 3D reconstruction in the previous tree. When searching for the edge $\mathbf{e}_{on}$, we only consider edges whose epipolar geometries are inconsistent with the previous spanning tree to skip trees leading back to the same 3D reconstruction.

We can keep the camera poses in T_l unchanged, and use a global transformation to update cameras in T_r by

$$\begin{bmatrix} \mathbf{R}^i_{new} & \mathbf{t}^i_{new} \end{bmatrix} = \begin{bmatrix} \mathbf{R}^i_{old} & \mathbf{t}^i_{old} \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & s \end{bmatrix}. \tag{3.8}$$

To decide $s, \mathbf{R}, \mathbf{t}$, we find graph edges that are consistent with the new spanning tree, i.e. $Prob(\mathbf{e}_{ij} \in S^c) < 0.4$, with one camera in T_l and the other camera in T_r. R is computed as the average of all relative rotations on these edges. We use corresponding 3D points reconstructed from T_l and T_r respectively to decide s and $\mathbf{T}$. At least two points are required for a unique solution. We follow (56) to select four reliable points on each candidate edge (this is done in the initialization stage for view pairs). We further check the reprojection error of these 3D points with the new camera poses. If the error is greater than 20 pixels, we discard the current $\mathbf{e}_{on}$ and search for the next.

Once the cameras are merged, we update the 3D positions of the reconstructed feature points. Recall that we have 3D reconstruction between each image pair during initialization. Given the camera poses, we use the baseline length to fix the scale of the pairwise reconstructions whose epipolar geometries are consistent with the new spanning tree. A feature point in an image has its depth reconstructed from multiple image pairs, each of which gives it a depth value. We sort all these depth values of each feature point, and choose the middle 20% values to compute an average depth for each image feature point. This approach to 3D reconstruction is highly efficient, since we only need to scale some existing pairwise reconstructions and average their resulted depths.

3.3.3 Fast Objective Function Evaluation

To make the evaluation of Equation (3.1) efficient, we give an alternative objective function definition as follows

$$E_F = \frac{1}{\sum_{i=1}^{n} m_i} \sum_{i=1}^{n} \sum_{p=1}^{m_i} \hat{P}_{missing}(p), \tag{3.9}$$

where m_i is the number of image features from view i with recovered depth (For computation efficiency, we divide the image into grid of cells with size 50×50 pixels and sample one feature from each cell). This objective function is slightly different from Equation (3.1). In fact, we can see

$$\sum_{i=1}^{n} \sum_{p=1}^{m_i} \hat{P}_{missing}(p) = \sum_{p=1}^{m} w_p \hat{P}_{missing}(p). \tag{3.10}$$

Here, w_p is the number of image features from which the 3D point p is reconstructed. Hence, besides the normalization factor, the difference between E_F and E_R is that E_F gives larger weights to 3D points associated with more image features. It is reasonable since these 3D points are more reliable. We also plot the values of Equation (3.9) in Figure 3.4 (b). The correct 3D reconstruction still corresponds to the global minimum of Equation (3.9). In fact, we prove the correct reconstruction should correspond to the global minimum of Equation (3.9) in Appendix A.

During the search of spanning tree, we need to compute the change in the new objective function in Equation (3.9) once $\mathbf{e}_{\mathit{off}}$ is removed or once $\mathbf{e}_{on}$ is added. To save computation, we do not compute Equation (3.9) from scratch. When $\mathbf{e}_{\mathit{off}}$ is removed, the drop in Equation (3.9) is equivalent to

$$\begin{aligned} E_D &= \tfrac{1}{\sum_{i=1}^{n} m_i} \sum_{i \in T_l} \sum_{p=1}^{m_i} \tfrac{1}{n} \sum_{j \in T_r} (P_{missing}(p,j) - \rho) \\ &+ \tfrac{1}{\sum_{i=1}^{n} m_i} \sum_{j \in T_r} \sum_{p=1}^{m_j} \tfrac{1}{n} \sum_{i \in T_l} (P_{missing}(p,i) - \rho). \end{aligned} \tag{3.11}$$

Intuitively, by removing the edge connecting T_l and T_r, points reconstructed from one subtree will become invisible in the images of the other subtree. Hence, we will replace their likelihood of inconsistency by the constant ρ. Further, the same term $P_{missing}(p, j)$ appears in the computation of E_D for different tree edges. We only compute each $P_{missing}(p, j)$ once and store its value for better runtime efficiency.

After the insertion of $\mathbf{e}_{on}$, we compute E_I, the increase in Equation (3.9) using the same expression as for E_D. Specifically, we update the probability of a point reconstructed in T_l (or in T_r) being missing in images in T_r (or in T_l). The energy of the new spanning tree is now given by

$$E_{new} = E_{old} - E_D + E_I. \tag{3.12}$$

3.3.4 Iterative search algorithm

To choose the two edges $\mathbf{e}_{\mathit{off}}$ and $\mathbf{e}_{on}$, we sort all edges on the previous spanning tree according to their drop in Equation (3.9) in descending order. We evaluate these edges one by one. For each edge, we look for $\mathbf{e}_{on}$ from the set of edges that are inconsistent with the previous spanning tree to link T_l and T_r. Once we find a pair $\mathbf{e}_{\mathit{off}}$ and $\mathbf{e}_{on}$ that lead to a E_{new} smaller than E_{old}, we remove $\mathbf{e}_{\mathit{off}}$ and add $\mathbf{e}_{on}$ to swap to a new spanning tree. The iteration stops when no such pair of $\mathbf{e}_{\mathit{off}}$ and $\mathbf{e}_{on}$ with lower energy can be found. We then use all the epipolar geometries consistent with the final spanning tree to compute the final 3D reconstruction with bundle adjustment. We summarize our algorithm in Algorithm 1.

Algorithm 1: Optimal spanning tree search.

```
Initialization:
1) Detect and match SIFT features to compute pairwise EGs. Keep SIFT
   features for fast objective function evaluation.
2) Sample a initial spanning tree on the match graph and compute camera
   poses with bundle adjustment.

Iterative search:
3) Classify epipolar geometries associated with match graph edges into
   consistent/inconsistent set according to the current spanning tree.
4) Sort tree edges according to $E_D$ in descending order.
5) Go through sorted tree edges one by one. For each $\mathbf{e}_{\mathit{off}}$, look for an $\mathbf{e}_{on}$
   from the inconsistent set, and evaluate the change of objective function.
6) If the objective function can be reduced, replace $\mathbf{e}_{\mathit{off}}$ by $\mathbf{e}_{on}$ to get a new
   tree and go to step 3)
7) If no result with lower energy can be found, stop.
```

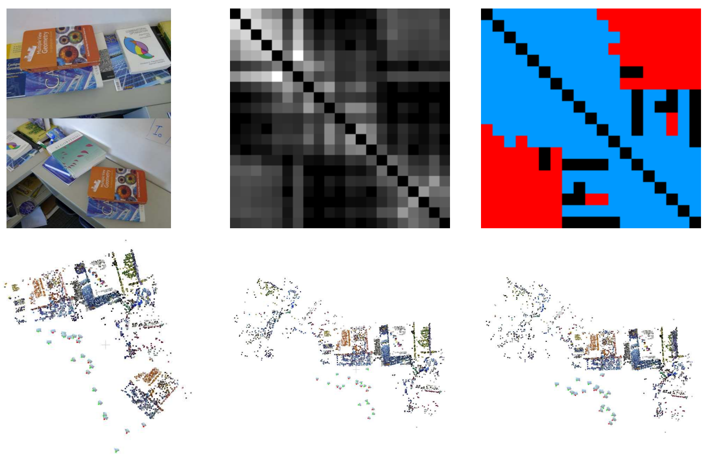
3.4 Experiments and Discussion

3.4.1 Experiments

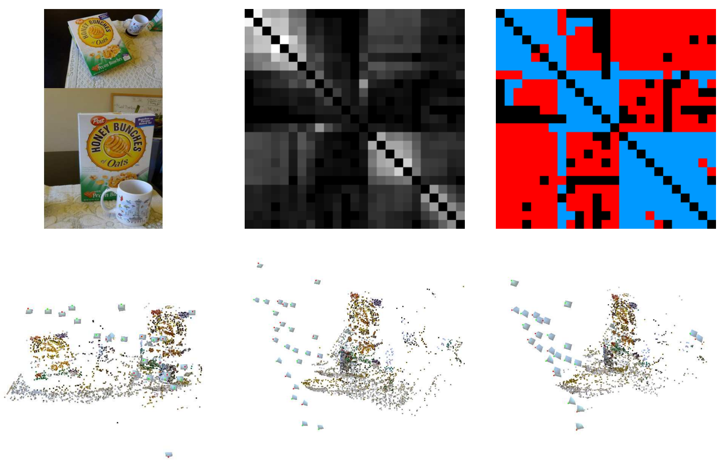
We experimented on a PC with Intel-Core2 Quad CPU that runs at 2.83GHz and 4GB RAM. We evaluated our algorithm with eight data sets as shown in Figure 3.5, Figure 3.6 and Figure 3.7 (final bundle adjustment is performed). In each row, the first three columns are two of the input images, weight matrix of the match graph, and binary labeling of the consistent (blue) and inconsistent (red) epipolar geometries upon convergence respectively. The last three columns are the visualizations of our results, the results from (99) and (83) respectively. As can be seen from the figure, (83) failed on all examples. (99) failed on all examples except the 'Desk' example in (d) . In comparison, our method can generate correct reconstruction among all these examples. Note that we only compare with (99) and (83) here, since their implementations are publicly available online. In fact, the examples (a)-(f) are from (74). As reported in (74), their method failed on (b), (c) and (e) when timestamps information was not used. Figure 3.6 (g) and Figure 3.7 (h) shows two additional examples with 153 and 150 input images respectively. Both of them have a large number of repetitive features. The cameras are incorrectly reconstructed at one side by (99) and (83). In comparison, our method generated good results on both of them.

We further provide the runtime efficiency for these algorithms in Table 3.1 (for all the methods we list both the runtime without/with final bundle adjustment, but exclude the computation of individual epipolar geometries). These examples are sorted in the same order as in Figure 3.5, Figure 3.6 and Figure 3.7. Though (99) is faster than our algorithm when the match graph is relatively simple, it

(a)



(b)



(c)

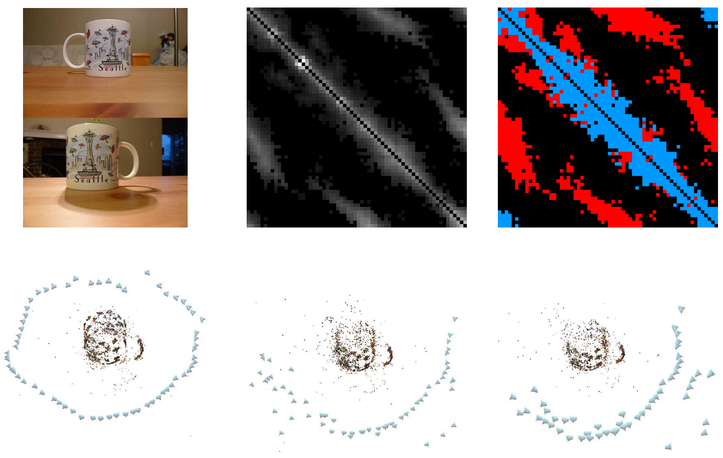
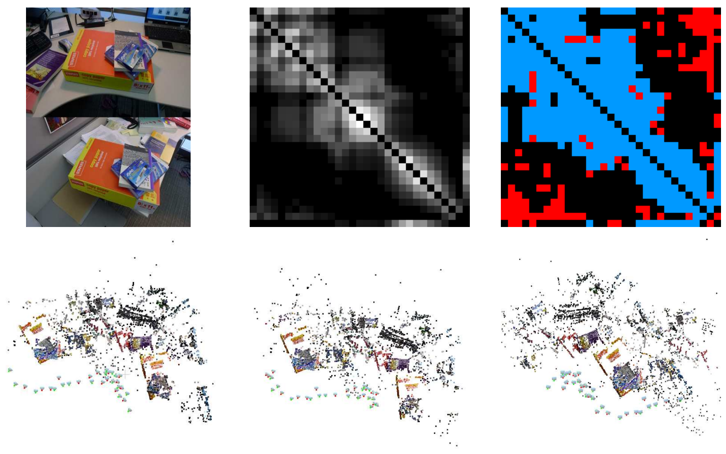
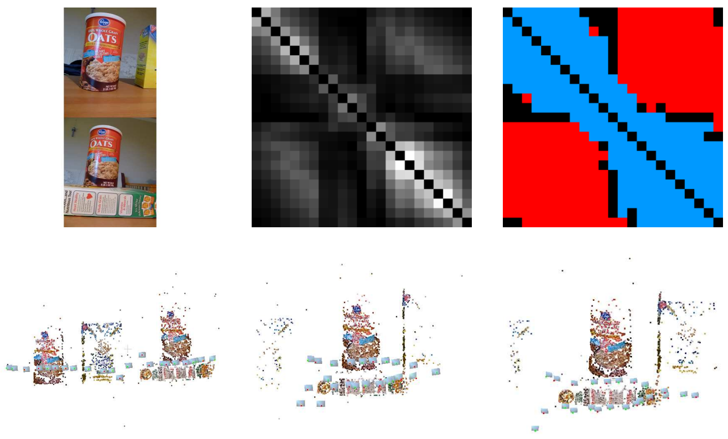


Figure 3.5: Experiment results on different data sets. For each example, from left to right in the first row are sample views from image sequence, weighted match graph, and binary labeling upon convergence; from left to right in the second row are 3D reconstruction using our algorithm, (99), and Bundler (83).

(d)



(e)



(f)

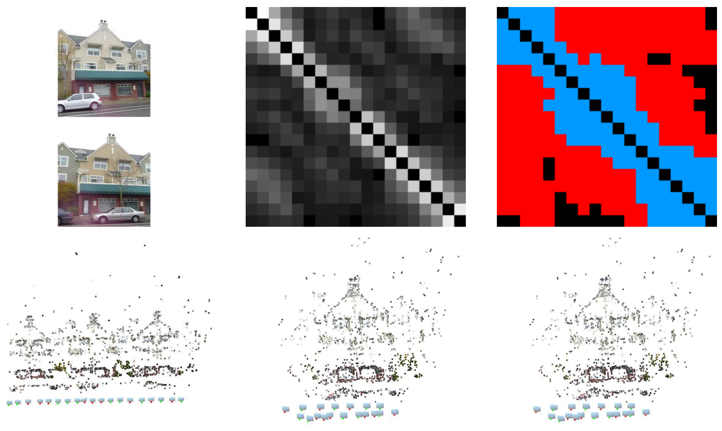
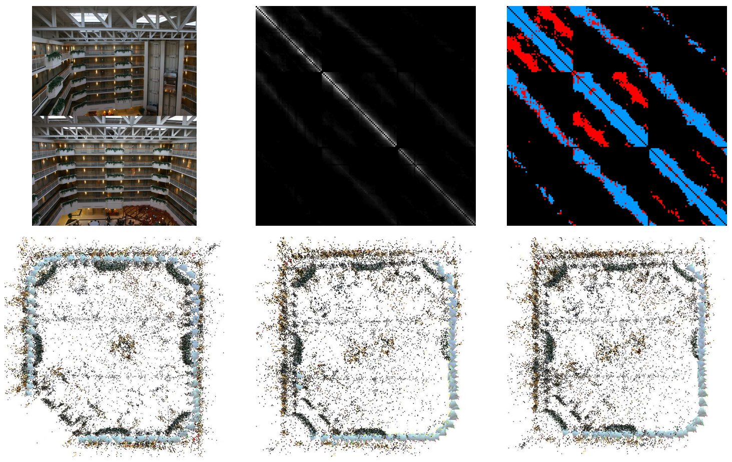
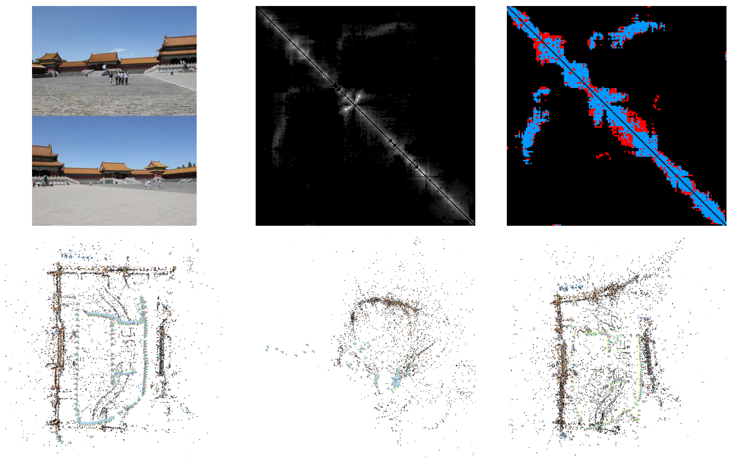


Figure 3.6: Experiment results on different data sets (continued from previous page and continued in the next page). For each example, from left to right in the first row are sample views from image sequence, weighted match graph, and binary labeling upon convergence; from left to right in the second row are 3D reconstruction using our algorithm, (99), and Bundler (83).



(g)



(h)

Figure 3.7: Experiment results on different data sets (continued from previous page). For each example, from left to right in the first row are sample views from image sequence, weighted match graph, and binary labeling upon convergence; from left to right in the second row are 3D reconstruction using our algorithm, (99), and Bundler (83).

Dataset	N	t_1	t_2	t_3
BOOKS[3]	19	37/72	919/928	1440/1740
BOXES	25	102/176	15/25	1680/1980
CUP	64	625/826	202/240	2640/3000
DESK	31	92/153	1869/1889	1800/2100
OATS[3]	23	59/114	1715/1740	1620/1920
HOUSE	19	19/49	6/9	2400/2700
INDOOR	153	1569/2707	369/424	-
FC	150	1792/2533	531/561	-

Table 3.1: Comparison of runtime efficiency. N is the number of input images. t_1, t_2 and t_3 are runtime (seconds) of our algorithm, (99) and (74) respectively.

often generates incorrect result. The running time of (74) was provided by the authors and obtained on a PC with a Core 2 Duo 3 GHz processor and 4GB RAM. They are much slower than our current implementation. The bottleneck of our algorithm is the evaluation of the objective function. This step could be easily parallelized to achieve significant speed-up for large scale data.

3.4.2 Discussion

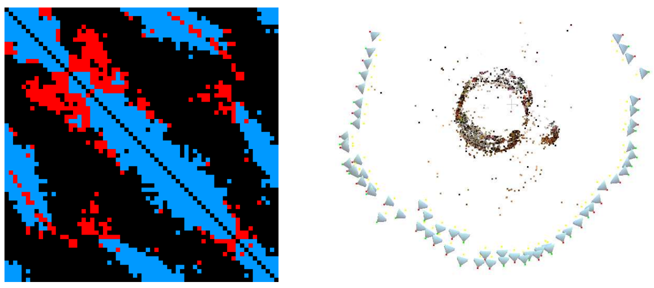
Convergence During the spanning tree search, we begin from the minimum spanning tree obtained on the weighted match graph. In our experiments, this minimum spanning tree often contains only a few (1-2) incorrect epipolar geometries. From such an initial tree, our method converged to the correct 3D reconstruction after traveling through 2-3 spanning trees. To test the capability of our greedy search algorithm, we deliberately chose initial spanning trees with larger number of incorrect epipolar geometries. We did this on the example in Figure 3.6 (f) by beginning with a randomly sampled spanning tree. We observed

[3]The duplicate objects in these sequences are created artificially by moving them around. We remove images with large portion of the duplicate object missing to prevent the discrepancy that will arise otherwise.

that the algorithm still found the correct solution after traversing 10-20 spanning trees starting from an initial one with 5-8 erroneous edges out of 18 in total.

Limitations We noticed mainly two limitations for our algorithm. First, the greedy search could get stuck at a local minimum. In our algorithm, we implicitly assume that, given an incorrect spanning tree, one can always find a tree with a lower score of Equation (3.9) by replacing ONE edge. This is however not true in general. Such an example is given for the 'cup' example in Figure 3.8 (a). Its final spanning tree has two incorrect EGs and cannot be improved by our algorithm. In other words, our method cannot guarantee to find the global minimum, though its convergence is guaranteed. Hence, in practice we might need to start from multiple different initialization, and choose the result with the minimum score in Equation (3.9). Second, our algorithm will fail on scenes with duplicate structures but little background features, such as the example in Figure 3.8 (b). This 'Temple of Heaven' example is rotationally symmetric. There are few 'background' points in the image. Hence, we cannot identify 'missing correspondences', and all the cameras are incorrectly reconstructed at one side of the building by our method.

In conclusion, we propose a method for robust structure-from-motion in scenes with large number of incorrect epipolar geometries, mainly caused by repetitive scene structures. We define a non-negative quantitative measure for the quality of a 3D reconstruction based on the idea of 'missing correspondences'. We show this function will attain global minimum for the correct 3D reconstruction. Hence, we design a greedy iterative algorithm to search for the correct 3D reconstruction by minimizing this function. For efficient search, we cache visited solutions and


(a)

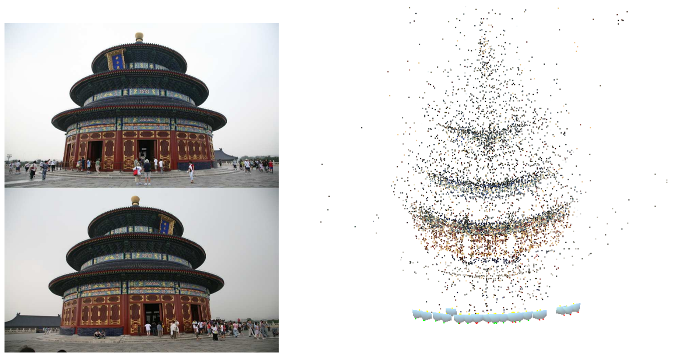

(b)

Figure 3.8: Failure cases for our algorithm.

revise the objective function to allow reuse of computation in previous iterations. The result is an efficient structure-from-motion algorithm that works robustly in highly ambiguous scenes.

Chapter 4

Joint Repetitive Structure Detection

4.1 Symmetry Detection

4.1.1 Overview

Symmetry detection is an extensively studied topic in computer vision. Symmetry information can be utilized for data completion, refinement or compression in 3D reconstruction and 3D modeling (5, 10, 60, 105). One of the most prominent symmetry property of architectural objects is the existence of repetitive structure elements, such as windows and balconies. Other than texture, the regularity of these structures in digital model is an important criteria on model quality assessment. To detect these repetitive structures automatically is a difficult problem and many methods are proposed in the literature.

Most existing works focus on detection of planar patterns from a single 2D

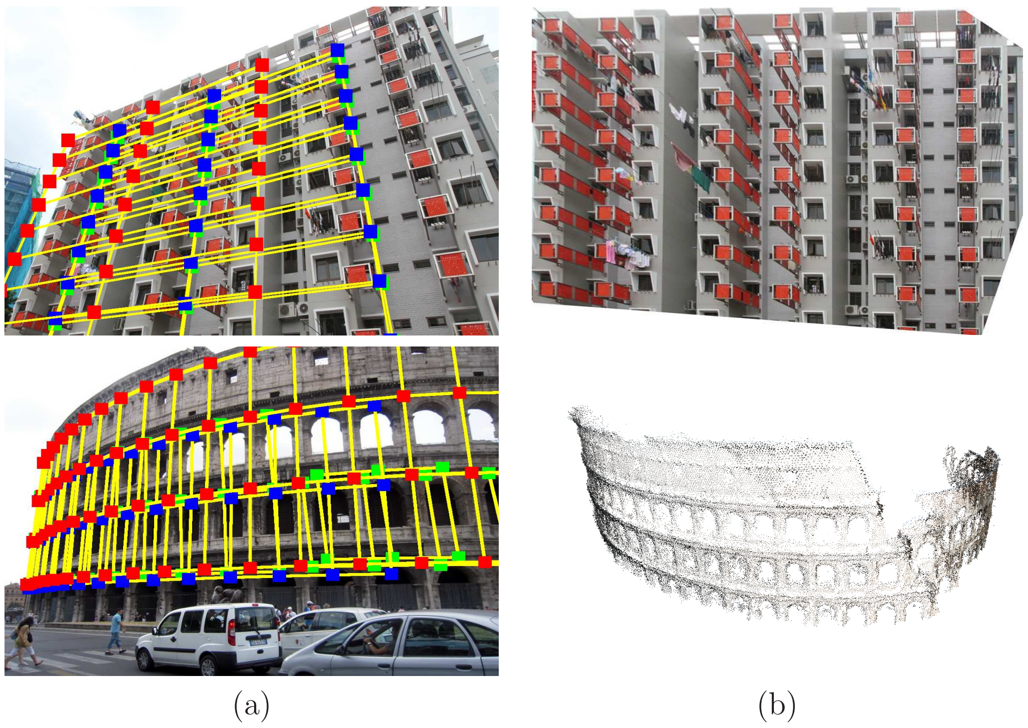


Figure 4.1: (a) Lattice detected by our method with curved surfaces and non-planar repetitive elements. (b) Top: the rectified image. The non-planar repetitive element makes the rectified image asymmetric. Bottom: the reconstructed 3D points are sparse and noisy.

image. There is a series of works, e.g. (32, 45, 46, 50, 52, 67, 68, 95, 103), to categorize and detect symmetries. When the repetitive structure lies on a curved surface, the detection is complicated by the deformation of repetitive elements and their lattice structure. To handle this problem, Hays et al. (32) and Park et al. (67) iteratively rectify the surface and detect a lattice structure in the rectified surface. However, this simultaneous estimation of deformation and lattice structure leads to complicated optimization. It is also difficult to apply them to non-planar 3D repetitive elements.

As mentioned before, real buildings often contain 3D repetitive structures such as balconies and windows. These repetitive structures can lie on curved building façades, which makes the detection even harder. Some examples of such buildings are provided in Figure 4.1. Focusing on these challenging data, we study repetitive structure detection from multiple images of the same scene. We employ the SfM algorithm described in Chapter 3 to reconstruct 3D point cloud from these images. There are two naïve ways for repetitive structure detection based on our input. First, we might rectify these images and detect repetitive pattern in the rectified picture with conventional methods. However, as shown at the top of Figure 4.1 (b), the non-planar repetitive elements (e.g. the red balconies) could make the rectified image asymmetric. Second, we might apply 3D symmetry detection methods, e.g. (11, 57, 70), to the reconstructed 3D points. However, our points are too sparse and noisy, as shown at the bottom of Figure 4.1 (b), to apply these methods, which require local geometric features such as surface curvature.

Hence, we propose to jointly analyze the reconstructed points and the multi-view images for repetitive structure detection. We first identify repetitive 3D points according to their image appearance in multiple views. We use 3D points to initialize repetitive structure hypotheses and verify them in images. Specifically, we estimate the underlying surface of these points by assuming that they can be described by a ruled quadric model, and rectify it to a plane to facilitate the analysis. Note that after we rectify the curved surface, like in the case of the Rome Colosseum example, the original rotational symmetry becomes a translational symmetry. As such, we only consider points that are related by translations or reflections in the rectified surface. This treatment of rotational symmetries is more general than that in (70), which estimates a 3D rotation axis and an angular interval and cannot handle elliptic cylinders like the Rome Colosseum example.

The detected repetitive structure can help us to enhance the quality of the reconstructed 3D points, which can benefit image-based modeling works such as (96, 97).

4.1.2 Related Works

2D Symmetry Detection Most of the symmetry detection algorithms such as (45, 50, 51, 54, 79, 84, 103, 104) focused on planar patterns. These methods can be regarded as local or global according to their methodologies. Local approaches like (51, 54, 79) extract a sparse set of corresponding features and hypothesize symmetry foci from pair-wise matches. These symmetry foci are then identified either via some voting schemes in a Hough transform fashion (54, 79) or exhaustive search in the parameter space (51). Global approaches use autocorrelation (50), the Fourier transform (45), co-occurrence matrices (84), or similarity map computed in scale space (103, 104) for discovering periodic patterns. All these methods share a common disadvantage in that both the repetitive elements and the underlying surface of these elements are assumed to be mostly flat and fronto-parallel in the image. Hence, they can hardly be applied to general architectural images taken from arbitrary viewpoints.

When a planar pattern is imaged from a slanted viewpoint, there is significant foreshortening effect. Cornelius and Loy (12) proposed a method to detect planar bilateral symmetry under such kind of perspective distortions. Wu et al. (95) rectified images according to vanishing points to facilitate repetitive structure detection. It is more challenging when the repetitive pattern lies on a curved surface, which causes spatially variant deformation. Hays et al. (32) iteratively rectified and estimated the topological lattice. This was further extended in (67) with the mean-shift belief propagation method to optimize the position of all lattice grids together. However, these methods require complicated optimization. Zhang et al. (106, 107) exploits the low rank property of image data to rectify slant textures or unwrap textures on generalized cylindrical surfaces. Proper initialization is required for correct rectification of the texture. Distraction from non-repetitive background and severe occlusion could fail the algorithm. Further more, no lattice structure is recovered here. In comparison, we utilize multiple images and 3D information from multi-view reconstructions for non-planar repetitive structure detection.

3D Symmetry Detection There are also a number of methods to detect symmetry in 3D data. Pauly et al. (70) and Mitra et al. (57) estimated the symmetry of dense laser scanned 3D data by analyzing its geometric signatures such as curvatures and tangent coordinate systems. In comparison, Bokeloh et al. (4) designed a novel 'line features' for symmetry detection. In a recent work, Bokeloh et al. (5) further applied the detected symmetries for inverse procedure modeling. Combes et al. (11) computed the symmetry plane of bilateral objects from laser scanned point clouds. Thrun and Wegbreit (87) searched for symmetries based on a hierarchical generate-and-test procedure. All these works require dense 3D point clouds for symmetry detection. Though we reconstruct 3D points from multi-view images, our data are much sparser and noisier, which makes these methods unsuitable. By utilizing rich texture information provided by multiple images, we can overcome the problem of sparse 3D points.

Symmetry-based Architecture Modeling Our work is also related to methods that exploit repetitive structures to facilitate architecture modeling. Müller et al. (59) analyzed the window patterns on a façade plane to generate its detailed 3D model. Korah and Rasmussen (42) detected and removed occluding objects from images by repetitive pattern analysis to generate clean texture maps. Nan et al. (60) and Zheng et al. (105) employed interactive methods to identify repetitive structures in laser scanned points for architecture modeling.

4.2 Joint Repetitive Structure Detection - the Algorithm

4.2.1 Algorithm Overview

Starting from multiple images of the same scene, we first apply structure-from-motion algorithm proposed in Chapter 3 and PMVS (47) to obtain a cloud of 3D points. Typically, we get about $50,000$ visible 3D points in each image (of resolution 1200×800). An example of this reconstruction is shown in Figure 4.1 (b).

We first identify multiple groups of repetitive 3D points to estimate the underlying curved surface (See Section 4.2.3). We rectify this surface to a plane to eliminate the geometric deformation of the underlying lattice structure. The appearance variation of repetitive elements is implicitly handled by the SIFT feature descriptor which is more robust to variations than the NCC approach in (32, 67) and by the availability of multiple images from different viewpoints. We identify a lattice structure for each group of repetitive points, and then cluster

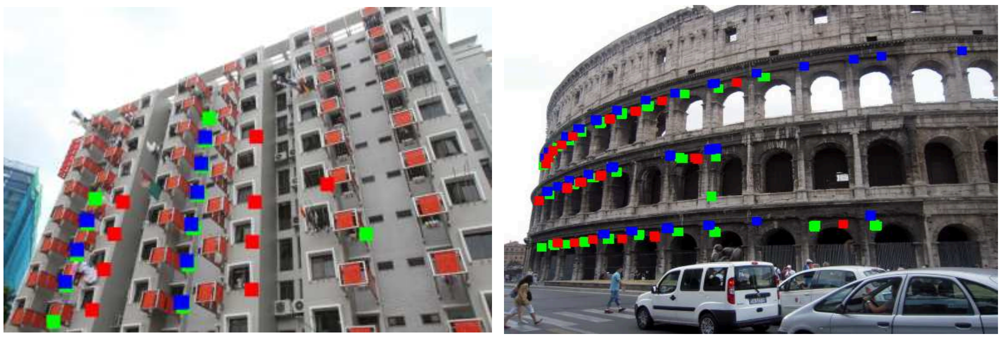


Figure 4.2: Detected repetitive points. Different groups of repetitive points are visualized in different colors.

and merge these results and report the most dominant one for each surface (See Section 4.2.4). Detected repetitive structure might be applied to clean up the reconstructed point cloud as in (60), which is helpful for image-based modeling applications (See Section 4.3). Experiments on real data and comparison with existing work (67) are provided in Section 4.4.

4.2.2 Repetitive Points Identification

We use the SIFT features (53) already extracted for 3D reconstruction to find repetitive points. We associate each reconstructed 3D point with features in multiple images where it is reconstructed from. We exhaustively check all pairs of SIFT descriptors associated with different 3D points. Repetitive points are identified if the angle between their descriptors is smaller than a threshold θ_1 (we set the threshold empirically as 20 degrees).

We consider the matching of repetitive points as an equivalence relationship. In other words, if two points both match with a third point, we also consider these two points as matched repetitive points. At the end of this step, we have repetitive 3D points in different groups according to their image appearance.

Some of the detected repetitive points are shown in Figure 4.2. Points from the same group are marked in the same color.

4.2.3 Structure Estimation

These repetitive points could lie on a curved surface, which causes geometric deformation of the lattice structure and complicates the detection. We recover and rectify this surface to facilitate the detection. We assume this surface is either a plane or a ruled quadric, which is true for most real buildings. The parametric model for plane is given by $n_1x + n_2y + n_3z + d = 0$, where $(n_1, n_2, n_3)^T$ is the normal of the plane. A quadric is denoted by a 4×4 matrix $\mathbf{Q}$, and any point on this quadric must satisfy $\mathbf{X}^T\mathbf{Q}\mathbf{X} = 0$.

We apply sequential RANSAC (19) to fit either quadrics or planes to each group of 3D points. In each pass, we select the model with most inliers. We summarize our model selection procedure as follows.

1. Apply RANSAC to the nine-point linear algorithm for quadric estimation. If the total number of points in the group is less than nine or all sampled points do not pass the degeneracy testing, go to step 3, else go to step 2.

2. The quadric with most inliers is estimated and converted to its canonical form. We further classify it into ruled quadric, degenerate quadric and general quadric based on rank estimation. If the quadric is a ruled quadric we add it to the model candidates, otherwise, go to step 3.

3. Apply RANSAC to the three-point plane estimation algorithm. If the total number of points is less than six, we exit from the sequential RANSAC for

the current group of 3D points; otherwise, add the plane with most inliers to the model candidates.

4. Select the model with more inliers and remove its supporting inlier points from the next round of model selection. Repeat step 1.

Different point groups (e.g. different corners on the repetitive balconies) often lie on similar surfaces that differ from each other by a small translation. We cluster these groups together. Ruled quadrics and planes are clustered separately. For this clustering, we simply stack all the 16 elements in $\mathbf{Q}$ or the normal of the planes to characterize a group. Two groups are clustered together if their 3D point constellations are close in space and their normalized parametric model vectors span an angle less than θ_2. (We fix it at 2 degrees in our implementation.)

The surface fitting can then be refined from multiple groups. Suppose the groups $g_1, g_2, \cdots, g_N$ are clustered together. We refine the surface $\mathfrak{S}$, a ruled quadric or a plane, by minimizing the following objective function

$$\sum_{i=1}^{N} \sum_{p \in g_i} R(\mathbf{p} - \mathbf{d}_i, \mathfrak{S}). \tag{4.1}$$

Here, $R(\mathbf{X}, \mathfrak{S})$ is the algebraic distance between a 3D point $\mathbf{X}$ to the surface $\mathfrak{S}$. The vector $\mathbf{d}_i$ is a translation in 3D space, which allows the surface of different groups to differ from each other by a translation. This minimization is solved in an iterative fashion. In each iteration, we first fix all $\mathbf{d}_i$ to estimate $\mathfrak{S}$ and then fix $\mathfrak{S}$ to estimate $\mathbf{d}_i$ for each group respectively. Both estimations only involve a linear equation and the whole process converges quickly. We begin this iterative fitting by letting $\mathbf{d}_i$ equal to zero. Some surface fitting results are illustrated in Figure 4.3. These surfaces are then rectified to a plane to facilitate the analysis.

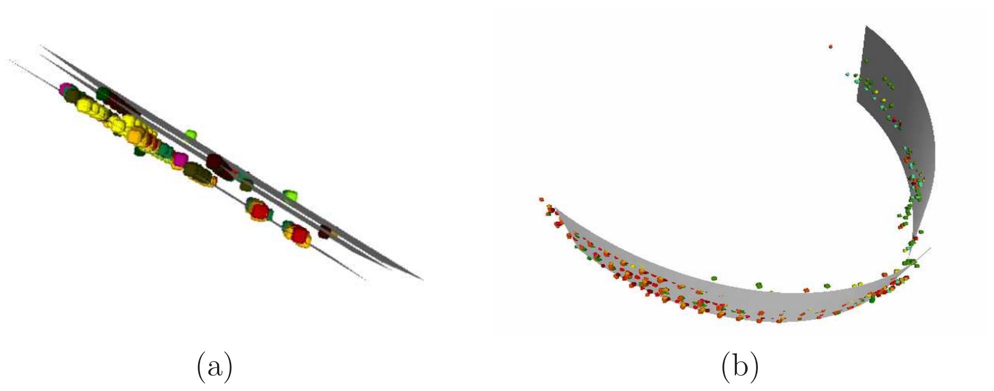


Figure 4.3: Surface fitted to multiple groups of repetitive 3D points.

4.2.4 Translational Lattice Detection

We first detect the underlying lattice for each group of points. We then consolidate these results to choose the most reliable parametric model for all groups $g_1, g_2, \cdots, g_N$ that share the same surface $\mathfrak{S}$.

Lattice initialization The 2D lattice structure is characterized by its two basis vectors. In the rectified surface, we check all pairs of repetitive points within a group, and compute a translation between each pair. A naïve lattice detection is to select the highest two local peaks in the histogram of these translations as the basis vectors. However, its performance is poor because the reconstructed 3D points are quite sparse and noisy. We treat these local peaks as candidate basis vectors and verify them according to the images as detailed below. An example histogram is provided in the first row of Figure 4.4.

Lattice validation To verify a basis vector, we select a 3D point as reference. Multiple grid points can be predicted on the line passing through the reference point along the direction of that vector. We compare the SIFT descriptors of

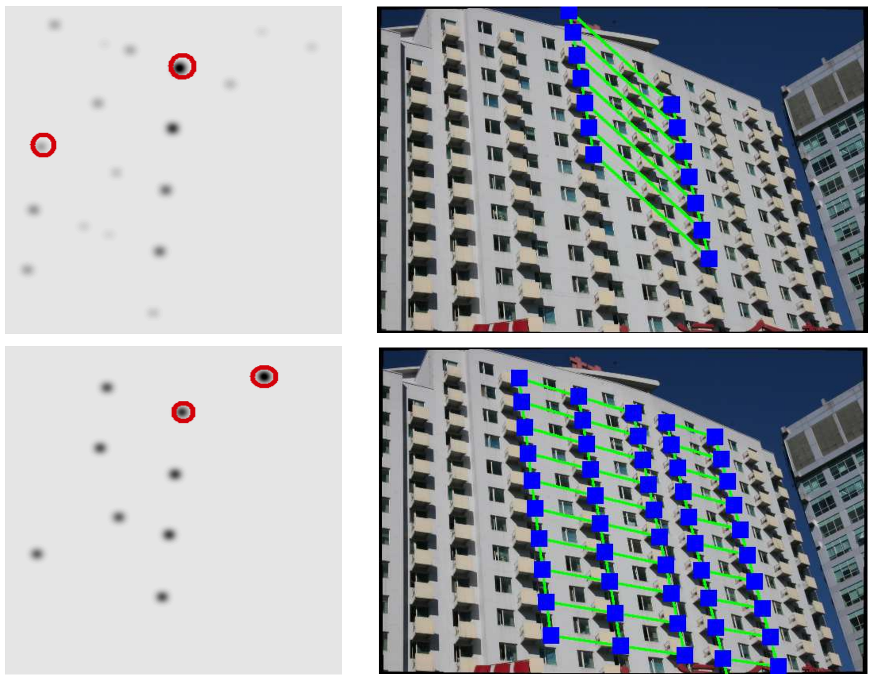


Figure 4.4: Top row: the histogram of pairwise translations and the detected lattice from this raw histogram. Bottom row: the image validation score of pairwise translations and the detected lattice from this score. In both rows, the two selected lattice basis vectors are circled in red. Note that in the original histogram space, one of the correct basis vectors has very low vote.

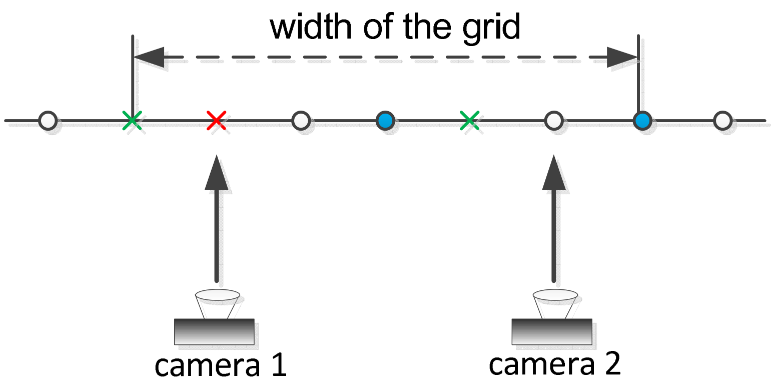


Figure 4.5: The red cross is a reference point. Blue and white circles indicate valid and invalid grid points. The SIFT descriptor of each point is obtain from its own most fronto-parallel image. The two green cross are the two farthest reconstructed points on the grid, which help to decide the width of the grid.

these grid points to that of the reference point. If the angle between them is smaller/larger than $2\theta_1$[1], we consider the grid point is valid/invalid. To handle appearance variation caused by foreshortening, the SIFT descriptor of a point is computed in its most fronto-parallel image, which is the one where the line connecting the camera center and that point is closest to the local orientation of the curved surface. This is illustrated in Figure 4.5, where the red crosses are the reference points, blue and white circles are valid and invalid grid points. The SIFT descriptors of the reference point and the grid points are computed from different images.

For each basis vector, we exhaustively check all reference points and define its image validation score as the total number of valid grid points. However, this score definition can be biased by directions with large number of repetitions, e.g. the vertical direction in tall building facądes. Hence, instead of using the number of valid grid points, we use the ratio between this number and the total number

[1]We use a looser threshold than the one in the detection of repetitive 3D points.

of grid points within a boundary. The sum of this ratio over all reference points is defined as the image validation score of a basis vector. To decide this boundary, as illustrated in Figure 4.5, We search for the two farthest reconstructed *on-grid* 3D points located on both sides of the reference point in the feature group. We consider a point to be on-grid if the distance between it and the nearest grid intersection is smaller than a threshold T_1 (10% of the basis vector length). Starting from these two initial points, we move them away from the reference point along the line until a significant portion T_2 (50% in our experiments) of the grid points within them are invalid. We then trim all the invalid grid points at both ends to obtain the grid's boundary.

After calculating an image validation score for all candidate basis vectors as shown in the second row of Figure 4.4, we can choose two of them to form the lattice structure. We sort these vectors in descending order of their lengths, and analyze them from top to bottom of the queue one by one. A vector with longer length is discarded if it can be represented as an integer combination of the rest of the queue. If two vectors are along the same direction, we only keep the one with higher score. Finally, we select two vectors with highest score from the remaining ones.

Lattice bundary estimation Once the two basis vectors are selected, we proceed to generate the lattice grids. The main challenge here is to decide a precise boundary of the lattice. We start from a 3D point and expand the grid by one row/column at a time, as shown in Figure 4.6. If the proportion of invalid points in an expanded row/column exceeds the significance threshold T_2, we stop and try to expand along the other directions. The lattice is finalized once all four

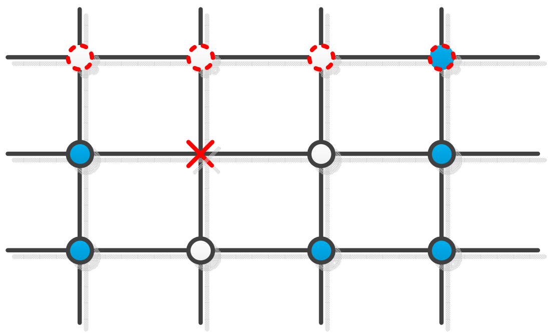


Figure 4.6: The red cross is a reference 3D point. Blue and white circles indicate valid and invalid grid points. The grid is extended one row in the top. Only one of the extended grid points is valid.

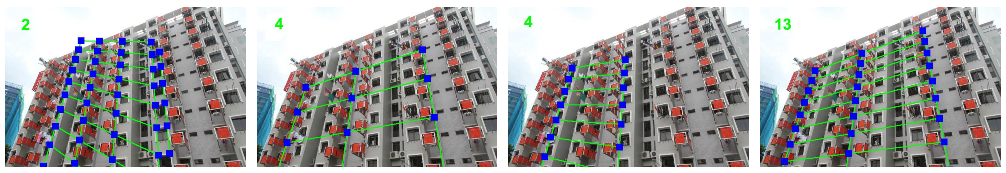


Figure 4.7: Clusters of grids of the same structure. The number of grids within a cluster is shown in the upper-left corner. The right most grid is selected from our consolidation.

directions cannot be expanded.

Lattice consolidation In real buildings, all the repetitive point groups on the same curved surface (e.g., different groups of repetitive corners on balconies) share the same lattice structure. Hence, we can consolidate the detection among these groups and generate one final result for each surface. We form multiple clusters of lattice structures. Two lattices are clustered together if the difference between their translation vectors is smaller than the threshold T_1. Among these clusters, we only keep the one with the largest number of lattices. An example of this consolidation is shown in Figure 4.7, where the biggest four clusters are shown. The number of grids in a cluster is shown at the upper-left corner. For this example, we finally choose the right most cluster.

4.2.5 Local Reflection Detection

After the lattice detection, we can look for local reflection symmetries on the lattices. Like (95), we only consider reflection symmetries in the 'vertical' direction, which is the direction of one of the two lattice basis vectors. We choose the one that is closer to the up direction of the input images.

Reflection axis estimation We exhaustively check all the vertical axes in the rectified surface. All 3D points used to fit this surface are used to vote for the right axis. For each candidate axis, we compare the SIFT descriptor of a 3D point with that of its mirrored point according to the axis. This pair of points is considered as valid if the angle between their SIFT descriptors is smaller than $2\theta_1$. Again, both descriptors are obtained from their most fronto-parallel images. We build a histogram of the number of valid pairs for all axes. If only a single dominant peak is found in this histogram (i.e. the second highest peak is lower than half of the highest one), we choose it as the reflection axis. Otherwise, there exist multiple valid axes. The horizontal interval between two neighboring axes should be the same as that between two lattice points (95). Hence, we fold the original histogram according to this interval, i.e. $\tilde{H}(k) = \sum_{i=k}^{i=k+T} H(i)$, where T is the interval, and find the strongest peak in the folded histogram to locate all these axes.

Symmetry boundary estimation For each detected symmetry axis, we set the boundary of the associated region as the bounding box of its valid point pairs. In the case of multiple repetitive symmetry axes, we compute a common boundary for all axes in the 'vertical' direction from their valid point pairs. Their width in the other direction is the same as the interval between axes.

4.3 Point Clouds Consolidation

Once the repetitive structure is identified, we can use it to enhance the point cloud like (60) to facilitate image-based modeling. Here, we only apply the translational symmetry to demonstrate this idea. We extract and align multiple blocks of 3D points according to the underlying lattice to generate more complete and denser results. All 3D points projected within a lattice cell form a block. Multiple blocks are extracted at different cells and aligned according to the lattice periodicity. We further apply the iterative closest point (ICP) algorithm (2) to refine the registration. Plane fitting and outlier removal could be applied subsequently. Figure 4.10 shows some point clouds before and after consolidation.

4.4 Experiments and Discussion

4.4.1 Experiments

We evaluated our method on images of different buildings with 3D non-planar repetitive elements. Some of the examples are provided in Figure 4.8. We used about 15 input images for each example[2]. As a rough average, our 3D reconstruction algorithm reconstructed about 100,000 points for each example and 50,000 visible points for each image, which is quite sparse compared with the image resolution, about 1200×800 pixels in our experiments. The first two columns

[2]This is not a guideline on the number of views that should be used. As long as reasonable reconstruction can be obtained, one can use as few as two views.

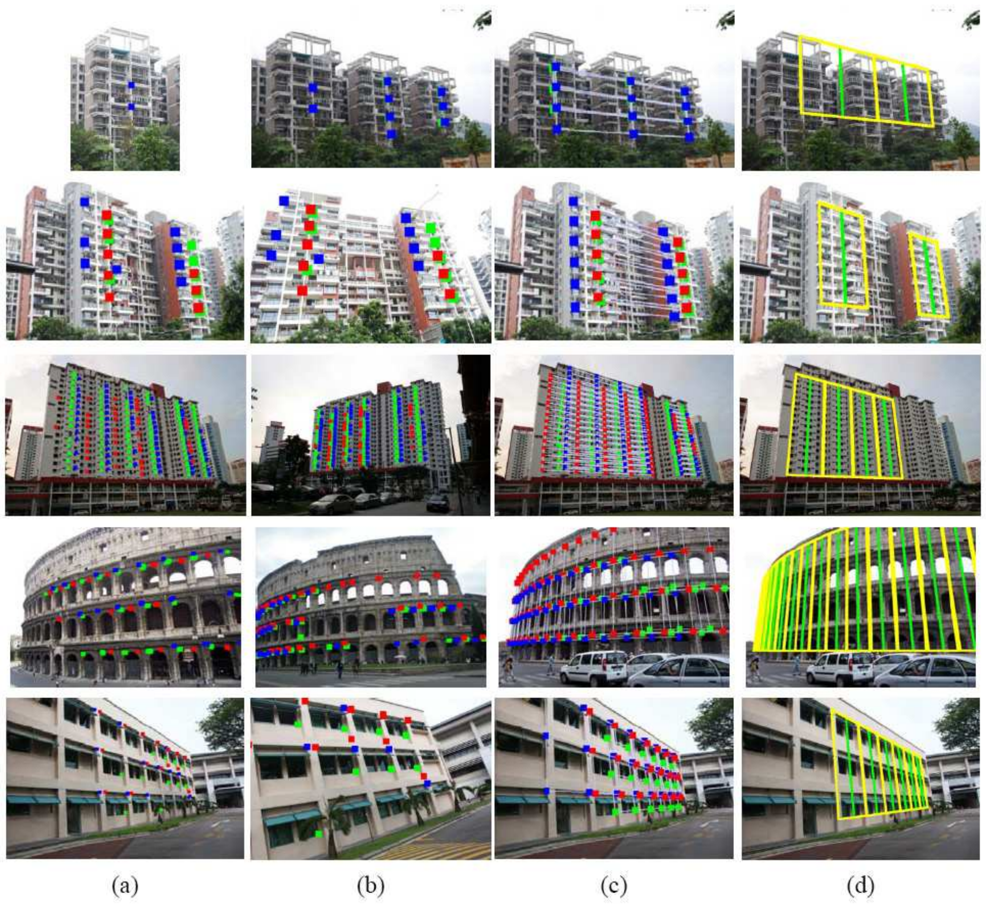


Figure 4.8: Results of repetitive structure detection. (a) and (b) are the left most and right most views of all input images. The detected repetitive points are overlaid on the image (the same group of repetitive points are visualized in the same color). (c) and (d) show the detected lattice and local reflection symmetry respectively.

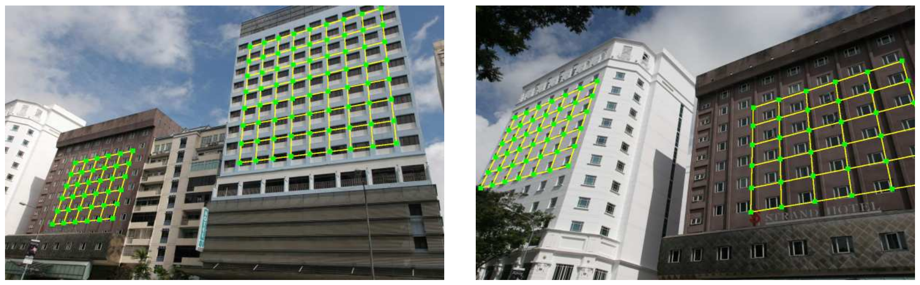


Figure 4.9: Lattice structures detected on multiple buildings. The images are two views from the same data sequence.

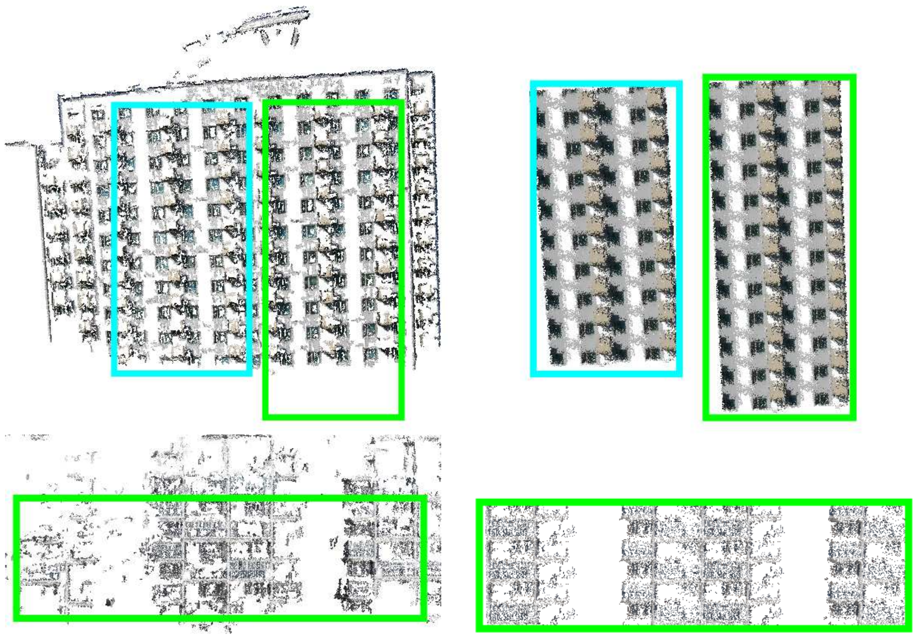

(a) (b)

Figure 4.10: Point clouds before and after consolidation. (a) shows the original points computed by our implementation of the structure-from-motion algorithm. (b) shows the consolidated point clouds tiled over the estimated grid. These two examples corresponding to the buildings in the third row in Figure 4.12 and the first row in Figure 4.8 respectively.

of Figure 4.8 show the left most and right most views of each building. The matched repetitive points are overlaid in these images, where points of the same color are from the same group. Typically, our program identified about 50 groups of repetitive points on each example. The detected lattice is shown in (c). For each example, we provide a few (1-3) different lattice that share the same basis vectors. In these examples (especially the first three rows), the repetitive elements are clearly non-planar. Yet our algorithm still correctly identified the lattice. The detected reflection symmetry is visualized in (d). The reflection axis is shown in green and the boundary is indicated by a yellow box. Note that our method works for images with multiple buildings, see Figure 4.9. Furthermore, since we apply RANSAC sequentially for surface detection within a repetitive point group, repetitive structure on multiple similar buildings can also be detected, e.g., the first example in Figure 4.12.

Additional results are reported in Figure 4.12. In these examples, (a) shows one of the input image with detected repetitive points. (b) is the estimated surfaces. To demonstrate the potential in image-based modeling, we further manually create a mesh for one repetitive element according to its consolidated 3D point cloud[3]. This mesh is then tiled over the lattice to generate the result shown in (d).

Comparison with (67) We compared our method with (67) on 16 different scenes with 373 images in total. We used the code provided by the authors. Some detection results from both methods are provided in Figure 4.11. It is clear that (67) tends to fail when the repetitive element resides on a non-planar surface. We

[3]Note that this element could be automatically generated by applying methods like (9).

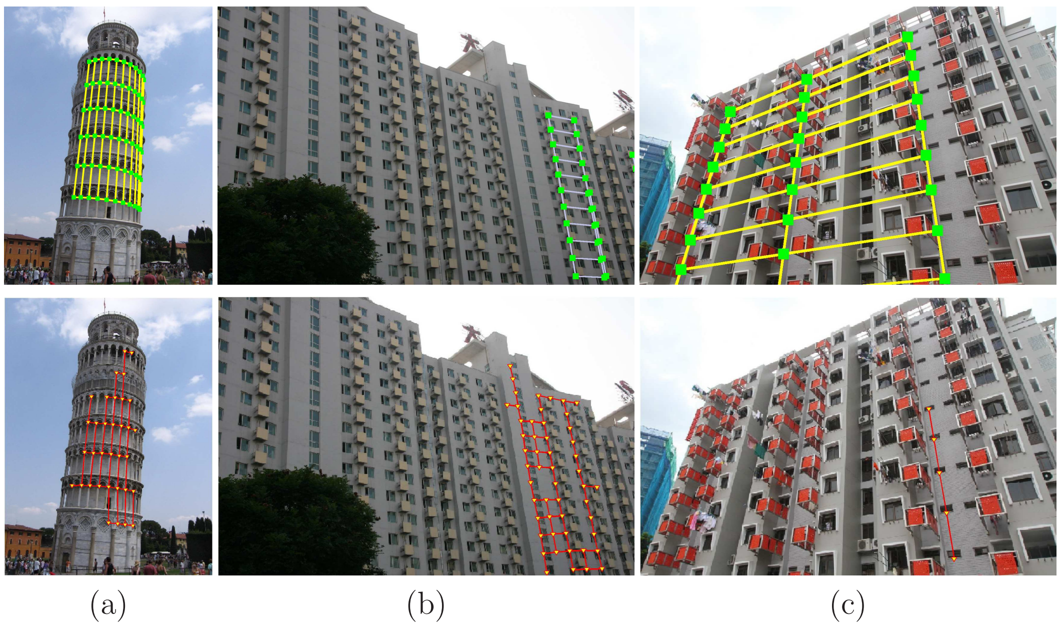


Figure 4.11: The first row is the lattice detected by our method, and the second row is that by (67). (a)both methods detected correct lattice. (b)our method detected partial lattice and (67) detected wrong lattice. (c)our method outperformed (67) in this case.

consider the detection a failure if a)no lattice is detected. b)wrong basis vectors are detected. or c)the detected lattice region is less than 30% of the actual one in the image. We evaluate the performance of both methods by two different counting rules: 1)use each image as a data sample. 2)use each sequence as a data sample, and score according to the best result in the sequence. Our method succeeds in 81 and 100 percent of these images, respectively. In comparison, the method in (67) can only handle 22 and 75 percent of the data, respectively (Refer to Appendix B for detailed results). We believe the strength of our method stems from the joint analysis of multi-view images and the reconstructed 3D points.

Comparison with (103) We further compare our method with (103) on an algorithmic level, since their algorithm deals mainly with fronto-parallel views but also detects translation symmetry in the rectified space. Given identified repeatitive points, Zhao et al. detect the translation bases by locating peaks in the transformed space using the breadth-first propagation. However, as illustrated in Figure 4.4, in the case where the transformation space is noisy and incomplete, the peaks in the raw transformation space coulde be incorrect. Without image verification of the hypothesized translation bases, which requires recovered 3D structure of the underlying surface, the detection of the translation bases could be unstable and erroneous. However, the MRF formulation of lattice generation proposed in (103) seems to work better with low-repetition patterns.

Point Cloud Consolidation To exemplify the point cloud consolidation, we provide examples before and after consolidation in Figure 4.10. It is clear that the original reconstructed points are much sparser with many holes. In comparison,

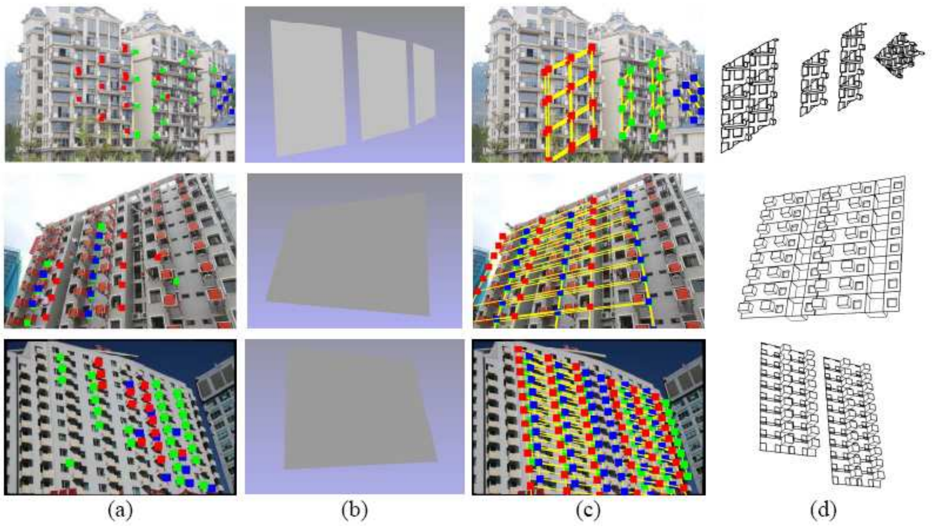


Figure 4.12: Additional results. (a) shows one of the input images with detected repetitive points (the same group of repetitive points are visualized in the same color). The underlying surface of these feature points is visualized in (b). (c) shows the estimated lattice. (d) is the 3D model of the surface.

the consolidated results capture the shape detail much better. These examples correspond to the buildings in the third row in Figure 4.12 and the first row in Figure 4.8 respectively. Please refer to their pictures to verify the geometric details.

4.4.2 Discussion

In conclusion, we present a method to detect architecture symmetries from multi-view images. Our method jointly analyzes these images and a cloud of 3D points reconstructed from them.

Parametric model fitting We fit quadrics or planes to these 3D points to initialize repetitive structure detections and verify these initializations according to images which contain dense color and texture information. The fitting works well most of the time, though it confuses shallowly curved surfaces with planes sometimes. The parametric model fitting approach is also not capable of handling general curved surface.

Local reflection symmetry In our proposed approach, we only consider local reflection symmetry that relies on the detection of lattice structures. However, there are many architectures that exhibit bilateral symmetry in a larger scale or on their overall shape. In fact, very recent work (10) adopt a similar approach that uses both image features and their associated 3D points to detect such large scale symmetry in the point clouds and use these additional information to perform constrained bundle adjustment for better reconstruction quality. Interestingly, the image pairs used for symmetry detection are exactly those causing visual ambiguity in structure-from-motion and classified as outliers in our work described in Chapter 3.

Other than the two major limitations, our method contains a number of thresholds, i.e. $\theta_1, \theta_2, T_1, T_2$, to decide if two SIFT features are similar, and if two groups are close to each other. However, since we apply them on normalized data, these parameters are easy to set and they are all fixed in our experiments. Thus, we conclude that the use of multiple views and joint analysis of 2D and 3D information make the algorithm capable of handling challenging data.

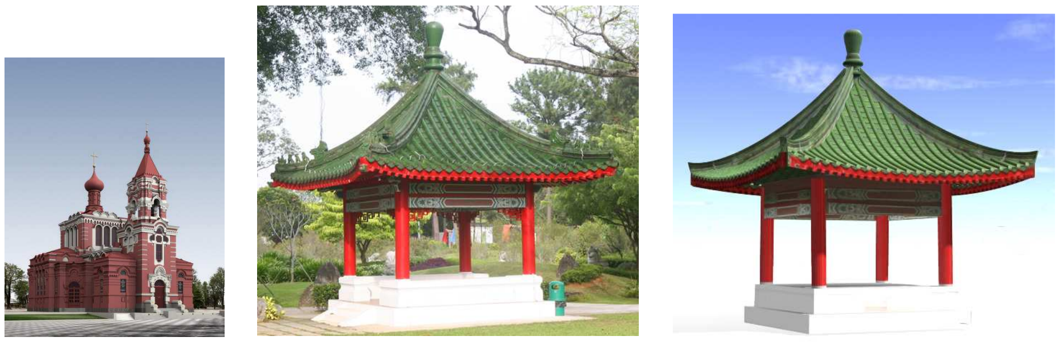
Chapter 5

Symmetry Assisted Architecture Modeling

5.1 Architecture Modeling

5.1.1 Overview

Creating high quality 3D architecture models is important for many applications including digital heritage, games and movies, etc. In addition to modeling softwares (such as Maya, 3DMax and Google SketchUp, etc), image-based modeling provides an alternative way to produce 3D mesh models. There are semi-automatic modeling systems, such as (16, 49, 58, 66, 81), which require user interaction; and also fully automatic modeling system such as (97). Despite the existence of various tools and systems available for 3D modeling, creating photo-realistic architecture models efficiently remains a challenging problem in computer vision and computer graphics. 3D architecture models created from commercial



(a) (b)

Figure 5.1: Architecture models created using modeling software and image-based modeling algorithm. (a) Church model created from 3dMax. (b) A pavilion modeled using our proposed modeling method. On the left is the single input image. On the right is the recovered model rendered from the same viewpoint as the input image.

modeling software often lack natural texture and look artificial, see Figure 5.1 (a). Creating 3D architecture models manually is also labour intensive and time consuming. Automated image-based model creation can generate photo-realistic street façades efficiently with little human intervention, but restricted to simple assumptions of the street layout (densely clustered buildings and little trees) and building geometry (planar façades). Interactive image-based modeling, on the other hand has the freedom of modeling complex 3D geometries and benefits from the efficiency of automatic 3D reconstruction techniques, Figure 5.1 (b) shows such an example created using our proposed modeling method.

Most of the previous works focus on piecewise planar architecture reconstruction and modeling from multiple images. Planar structures induce strong shape constraint and simplify the 3D modeling from point clouds. However, many traditional and more artistic architectures have intricate geometric structure and curved roofs, which are highly non-planar and cannot be modeled well by existing methods, Figure 5.2 (a) shows such an example. Yet, these buildings are

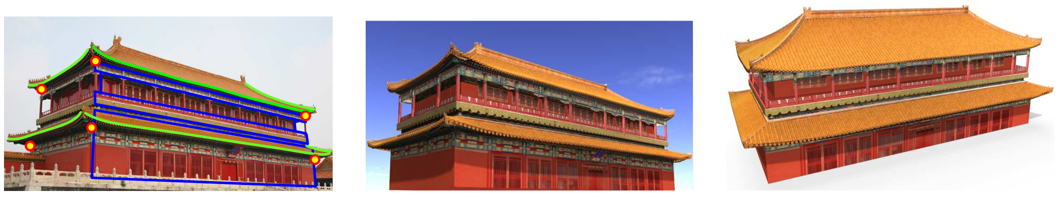

(a) (b) (c)

Figure 5.2: A traditional Chinese architecture, the Pavilion of Manifest Benevolence (also known as TiRen Ge) in the Forbidden City, is modeled from a single input image. (a) is the input image overlaid with user-drawn strokes. (b) is the rendering of the recovered model from the same viewpoint as the input image for validation. (c) shows the rendering from a novel viewpoint.

often landmarks that are particularly worthy of being modeled. Despite the fact that these buildings cannot be well described by piece-wise planar surfaces, there often exists strong geometric constraints on the overall building shape, e.g. symmetry. In this chapter, we advocate exploiting symmetries for 3D reconstruction and modeling. To push it to the limit, we study the problem of 3D reconstruction and modeling from a single image. Single image based modeling is difficult. First, it is difficult to calibrate the camera (i.e. recovering both intrinsic and extrinsic camera parameters), which is necessary to relate the image to the 3D model. Second, a single image often does not provide enough texture information due to foreshortening and occlusion. 3D modeling from a single image also has its practical importance since multiple images of the same building are not always available.

As Magdolna and Hargittai (28) have commented, symmetry is ‘a unifying concept’ in architecture. A single image of a symmetric building effectively provides observations from multiple symmetric viewpoints (21, 33, 102). In other words, shape symmetry effectively upgrades a single input image to multiple images. To exploit this property, we first propose a method to calibrate the camera from a single image according to the presented symmetry. This calibration allows our system to handle images with completely unknown camera information (e.g. internet downloaded pictures and archive pictures). Then, a virtual camera is duplicated at the position symmetric to the real camera, and its observed image is derived from the input image. A stereo algorithm follows to recover a set of 3D points from the real and virtual camera pair. After that, the user interactively organizes the reconstructed 3D points into a high quality mesh model. To keep the interaction simple, the user only manipulates in the image space to mark out various architecture components such as walls and roofs, whose shapes and positions in 3D are automatically computed. Symmetric counterparts of each marked component are generated automatically to reduce the user interaction. Thanks to the strong symmetry, the modeling process typically takes less than 5 minutes of interaction. Lastly, the model is textured according to the single input image. We use symmetry again to enhance the texture quality at those foreshortened and occluded regions.

In summary, we proposed a systematic architecture modeling method building upon the ubiquitous architecture symmetries. We build a novel camera calibration algorithm (Section 5.2.1), an efficient interactive architecture modeling interface (Section 5.3) and a practical texture enhancement method (Section 5.3.2). All these components prove architecture modeling can be made very efficient by making appropriate usage of symmetry-based constraints.

Figure 5.3 shows the pipeline of our system. We first calibrate the camera from a single image. Then we reconstruct a set of 3D points according to the calibration and architecture symmetry. Next, the user interactively marks out structural components such as roofs and walls to build an initial 3D model. The

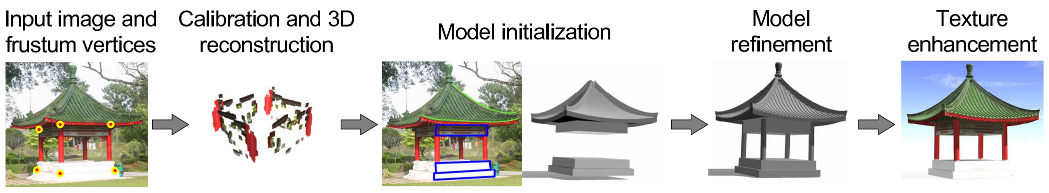

Input image and frustum vertices
Calibration and 3D reconstruction
Model initialization
Model refinement
Texture enhancement

Figure 5.3: The modeling pipeline. We first calibrate the camera according to the user specified frustum vertices and reconstruct a set of 3D points. The architecture components (i.e. walls and roofs) are then interactively decomposed and modeled. Shape details can be added if necessary. Lastly, the final model is textured with the texture enhancement technique described in Section 5.3.2.

user can also add in more geometric detail, such as roof tiles and handrails, or insert predefined primitives, such as pillars and staircases. At last, the recovered model is textured according to the input image. Texture synthesis is used to improve texture quality at the foreshortened and occluded regions.

5.1.2 Related Work

3D reconstruction and architecture modeling have received a lot of research interest, with a large spectrum of modeling systems developed to build realistic 3D models. Here we only review those works related to symmetry and architecture modeling. We categorize them according to their methodologies.

3D reconstruction from symmetries It is well known that symmetry provides additional constraint for 3D reconstruction. Rothwell et al. (76) and Francois et al.(21) studied the 3D reconstruction of bilaterally symmetric objects. Zhang and Tsui (102) extended it to handle arbitrary shape by inserting a mirror into the scene. Hong et al. (33) provided a comprehensive study of reconstruction from various symmetries. Most of these works focus on bilateral symmetry and study the resulting multi-view geometric structures such as the special configurations of the fundamental matrix and epipoles. These works often assume the camera is pre-calibrated with known focal length and/or pose (position and orientation) for 3D reconstruction. Although Hong et al. (33) studied the camera calibration problem from symmetries, their results are limited, e.g. the camera can be calibrated from the vanishing points of three mutually orthogonal axis of bilateral symmetry. In comparison, we focus on the application of architecture modeling and study both bilateral and rotational symmetries. Since we are more specific about the object to be modeled, we obtained stronger results both on camera calibration and texture creation, which lead to a complete system for high quality modeling with a single uncalibrated image.

Procedural architecture modeling Procedural methods build 3D architecture models from rules and shape grammars. They can generate highly detailed models at the scale of both an individual building and a whole city (58, 66). A disadvantage of these methods is that it takes expertise for its effective usage. It is also hard to specify rules to model a particular building.

Interactive architecture modeling Debevec (16) fitted a parametric building model to the single (or multiple) input image(s) according to the user marked geometric primitives. High quality results can be achieved. There are also commercial modeling systems like Google SketchUp, where the user sketches freely to create a 3D building model from scratch or according to an image. The major limitation of these two systems is the large amount of user interaction involved. In Google SketchUp, all the shape details have to be sketched manually. As reported in Debevec's PhD thesis (15), for the relatively simple Berkeley Campanile example (see Figure 5.14), about one hundred edges need to be manually marked and corresponded which takes 2 hours. Our method involves much less interaction, because the 3D information is explicitly recovered *before* the interactive façade decomposition and reconstruction. With our system, the user draws less than 20 lines and it takes only 9 minutes (for a novice) to model the Berkeley Campanile building. Another limitation of the Façade system is that it requires the camera to be pre-calibrated with known intrinsic parameters. To handle uncalibrated cameras, the vanishing points of three mutually orthogonal directions need to be detected from the image (15), which is often impossible (e.g. for buildings in Figure 5.11–Figure 5.13) and numerical unstable (94). In comparison, our novel auto-calibration algorithm is more robust and can handle more general data, which is a critical feature for a desktop modeling toolkit.

Single image based architecture modeling Images provide very useful information to assist modeling. Even a single image can guide the modeling quite effectively. Liebowitz et al. (49) created a 3D model by exploiting parallelism and orthogonality from a single image. Oh et al. (63) manually assigned a depth with a painting interface to create 3D model. Such a procedure is tedious and labor intensive. Müller et al. (59) derived shape grammars from a single image of a façade plane. These single image based methods are limited to simple buildings. While our method also takes a single image as input, we explicitly reconstruct 3D points from the input image, which helps both to simplify the user interaction and to model more complicated buildings.

Multiple images based architecture modeling Multiple images from different viewpoints provide strong geometric constraints on 3D structure. Dick et al. (17) built statistical model to infer building structure from multiple images. However, such inference is unreliable for complex buildings. The multi-view stereo algorithm developed in the computer vision community can generate cloud of 3D scene points from multiple images, which lead to more robust reconstruction. Sinha et al. (81) used an unordered collection of pictures to assist interactive building reconstruction. Xiao et al. (96) took pictures along streets and built 3D models of the whole street. Pollefeys et al. (72) developed a real-time system for urban modeling from video data. Our method is inspired by the work of Sinha et al. (81) and Xiao et al. (96), where reconstructed 3D points can be used to guide user for efficient interaction. Specifically, if 3D points are reconstructed, tedious manual correspondence as in (16) can be avoided. The user only needs to mark out structural components, whose shape and position can then be determined from the reconstructed 3D points. The availability of multiple images always yields better 3D reconstruction and modeling results, but we have to resort to single-view modeling when we only have one image as input.

Aerial images based architecture modeling There are also methods (100) which used aerial images to reconstruct buildings. Some of them (23) combine aerial images with ground-level images for the modeling. The focus of these methods is on how to efficiently model very large set of data. As such, the quality of each individual building could be sacrificed for modeling efficiency. In this study, we focus on how to create a high quality model for a single building.

The proposed method combines the strength of both interactive modeling and image-based modeling. We take a single image as input and reconstruct explicit 3D information by leveraging on prevalent architectural symmetry. The reconstructed 3D information helps us to design a more efficient interface than previous interactive methods and single image based methods. Compared with those methods with multiple images, our system is more flexible since it requires much less data.

5.2 3D Reconstruction by Symmetry

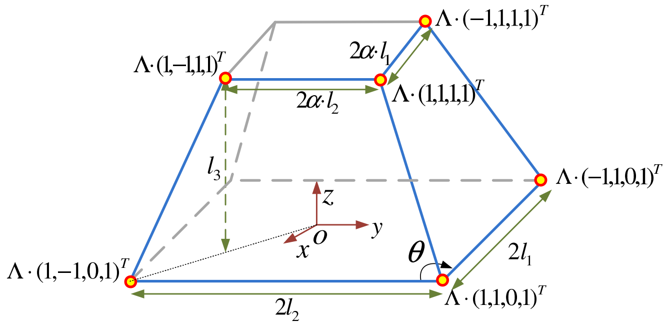
In this section, we describe how to reconstruct the camera pose and a set of 3D points from a single image by exploiting architectural symmetries, including both bilateral and rotational symmetry. We first calibrate the camera from an observed pyramid frustum. Then we duplicate a virtual camera according to the calibration and the observed symmetry. 3D points are computed by stereo algorithm from the real and virtual cameras.

5.2.1 Symmetry based Camera Calibration

Cameras need to be calibrated for undistorted 3D reconstruction. The calibration accuracy is important as the image is related to the 3D model according to the calibration.

The camera can be calibrated from the vanishing points of three mutually orthogonal directions in a single image (30), which is applied for façade modeling in (15). However, many images, e.g. Figures 5.11– 5.13, do not have three such vanishing points. Furthermore, the vanishing point based approach is often numerically unstable (94). Naturally embedding the constraints from three vanishing points, an parallelepiped in a single image can be used to calibrate the camera (93, 94). The details of the calibration technique is described in Chapter 2. This approach is stable and accurate and is applied for architecture modeling. The parallelepiped, however, is not the most suitable geometric primitive for architecture. A degree of freedom is redundant for architecture since the horizontal shearing of a parallelepiped is not present in real buildings. On the other hand, the horizontal size of real buildings often gradually shrink when the height increase. This feature is common in architectures, as illustrated in Figure 5.2 and Figures 5.11–5.13, but it cannot be represented by a parallelepiped. A better geometric primitive is the pyramid frustum, which does not introduce the redundant degree of freedom and can model real buildings well.

Parametric model of pyramidal frustum A pyramidal frustum is a truncated pyramid as illustrated in Figure 5.4. Here, we use a frustum with a rectangular base as an example for discussion, though our results are valid for frustums with different bases. We parameterize a pyramid frustum by $\alpha, \theta, l_1, l_2, l_3$, as illustrated in Figure 5.4. α is ≤ 1 and controls the shrinking of the pyramid. If $\alpha = 1$, the pyramid frustum degenerates to a right prism, a parallelepiped with zero horizontal shearing. θ is the angle between the two adjacent horizontal edges of the frustum base. $l_i, 1 \leq i \leq 3$, are the three independent lengths of the structure. For modeling applications, the absolute position and size of the structure is not important. Hence, without loss of generality, we can let the height $l_3 = 1$ and consider the origin of the world coordinate system to be at the bottom face of the frustum, with the z-axis passing through the apex of the pyramid, and the y-axis parallel to one of the edges. From a single image of a building, part of a pyramid



$\Lambda\cdot(-1,1,1,1)^T$
$2\alpha\cdot l_1$
$\Lambda\cdot(1,-1,1,1)^T$
$2\alpha\cdot l_2$
$\Lambda\cdot(1,1,1,1)^T$
l_3
z
y
o
x
$\Lambda\cdot(-1,1,0,1)^T$
θ
$2l_1$
$\Lambda\cdot(1,-1,0,1)^T$
$\Lambda\cdot(1,1,0,1)^T$
$2l_2$

Figure 5.4: A pyramid frustum is a truncated pyramid. It shape is defined by 5 parameters. $l_i, 1 \leq i \leq 3$ defines the length of its edges. α controls the shrinking in vertical direction. θ is the angle between the two horizontal edges. Blue edges and red vertices are the parts of a frustum that are often visible in architecture images.

frustum can often be seen (the highlighted vertices and edges in Figure 5.4). The corresponding points are highlighted in the Figure 5.2 (a).

Pyramid frustum and camera calibration The homogeneous coordinates of a frustum vertex can be represented as $\mathbf{X}_i = \mathbf{\Lambda} \cdot \hat{\mathbf{X}}_i^\top$, where $\hat{\mathbf{X}}_i = (x_i, y_i, z_i, 1)$, and $x_i, y_i \in \{1, -1\}, z_i \in \{0, 1\}$ (see Figure 5.4). Here,

$$\mathbf{\Lambda} = \begin{pmatrix} l_1 & l_2 c & 0 & 0 \\ 0 & l_2 s & 0 & 0 \\ 0 & 0 & \beta l_3 & 0 \\ 0 & 0 & \beta - 1 & 1 \end{pmatrix} \tag{5.1}$$

where $\beta = 1/\alpha, s = \sin\theta$ and $c = \cos\theta$. As $\mathbf{\Lambda}$ contains all the shape parameters of the pyramid frustum, the 3D reconstruction of the frustum amounts to the estimation of $\mathbf{\Lambda}$. Frustum vertices are projected into image coordinate $\mathbf{x}_i = (x_i, y_i, \lambda_i)$ by the projective transformation $\mathbf{P}$, i.e.

$$\mathbf{x}_i \simeq \mathbf{P}\mathbf{X}_i = \mathbf{P}\mathbf{\Lambda}\hat{\mathbf{X}}_i = \hat{\mathbf{P}}\hat{\mathbf{X}}_i \tag{5.2}$$

where $\simeq$ means equality up to a scale. $\mathbf{P} = \mathbf{K} \cdot [\mathbf{R}|\mathbf{t}]$ is the 3×4 camera matrix, where $\mathbf{K}$ encodes the camera intrinsic parameters, $\mathbf{R}$ and $\mathbf{t}$ represent relative rotation and translation between the camera and the world coordinate system. If six or more frustum vertices can be observed from the image, $\hat{\mathbf{P}}$ can be computed by a linear algorithm (30).

Camera calibration and 3D reconstruction of the pyramid frustum then amounts to the factorization of $\hat{\mathbf{P}}$ as

$$\hat{\mathbf{P}} = \mathbf{K} \cdot [\mathbf{R}|\mathbf{t}] \cdot \Lambda \tag{5.3}$$

A general camera intrinsic matrix $\mathbf{K}$ contains 5 unknowns. $\mathbf{R}, \mathbf{t}$ each contains 3 unknowns. Λ has another 4 unknowns (considering $l_3 = 1$), making a total of 15 unknowns. The 12 components of the 3×4 projective matrix $\hat{\mathbf{P}}$ provide only 11 independent constraints. This factorization is impossible without further assumption about the camera parameters and the scene structure. The assumption involves the trade-off between the generality of the camera model and the frustum structures. To model a larger variety of buildings, we assume the simplest camera model where only the focal length is unknown[1]. With this simplification, all the 11 unknowns can be computed from the 11 constraints with a general non-linear optimization method. If further information is known about the architecture structure as a prior, such as the value of the angle θ or the length ratio between l_1 and l_2, we can handle more general camera matrix with unknown pixel aspect ratio or principal point.

[1]The other known camera parameters are the principal point, the pixel aspect ratio, and the camera skew.

Quadratic initialization A good initialization is critical for the success of the above non-linear optimization. In this subsection we describe a method to initialize the estimation by solving a quadratic equation. We observe that

$$\hat{\mathbf{P}}^\top \mathbf{K}^{-\top} \cdot \mathbf{K}^{-1}\hat{\mathbf{P}} = \begin{pmatrix} l_1^2 & l_1 l_2 c & * & * \\ l_1 l_2 c & l_2^2 & * & * \\ * & * & * & * \\ * & * & * & * \end{pmatrix}. \tag{5.4}$$

Here, $\mathbf{K}^{-\top} \cdot \mathbf{K}^{-1} = \omega$ is the matrix representing the image of the absolute conic (see also Chapter 2). Hence, we have the following equations,

$$\hat{\mathbf{p}}_1^\top \omega \hat{\mathbf{p}}_1 = l_1^2; \qquad \hat{\mathbf{p}}_1^\top \omega \hat{\mathbf{p}}_2 = l_1 l_2 c; \qquad \hat{\mathbf{p}}_2^\top \omega \hat{\mathbf{p}}_2 = l_2^2. \tag{5.5}$$

Here, $\hat{\mathbf{p}}_1, \hat{\mathbf{p}}_2$ are the first two columns of $\hat{\mathbf{P}}$. Assuming the simplest camera model, ω depends only on the focal length f. Equation (5.5) provides 3 equations for 4 unknowns l_1, l_2, θ, f. From a single image, very often we can either tell the value of θ or the length ratio of l_1 and l_2, which reduces one unknowns from Equation (5.5) and enables the recovery of the other threes. This provides the initialization of l_1, l_2, θ, and f. Next, we initialize the other unknowns, i.e. $\mathbf{R}, \mathbf{t}$, and β.

Once f is determined, $\mathbf{K}$ is known and we can compute

$$\begin{aligned} [\mathbf{R}|\mathbf{t}]\Lambda &= \mathbf{K}^{-1}\hat{\mathbf{P}} = [\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{t}]\Lambda \\ &= [l_1\mathbf{r}_1, l_2 c_2 \mathbf{r}_1 + l_2 s_2 \mathbf{r}_2, \beta\mathbf{r}_3 + (\beta - 1)\mathbf{t}, \mathbf{t}]. \end{aligned} \tag{5.6}$$

Here, $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3$ are the three columns of $\mathbf{R}$. Hence, $\mathbf{r}_1$ can be obtained by normalizing the first column of $\mathbf{K}^{-1}\hat{\mathbf{P}}$. $\mathbf{r}_2$ is generated by projecting the second column to the plane perpendicular to $\mathbf{r}_1$, and $\mathbf{r}_3$ is simply $\mathbf{r}_1 \times \mathbf{r}_2$. $\mathbf{t}$ is the last column of $\mathbf{K}^{-1}\hat{\mathbf{P}}$. β can be obtained from the magnitude of the third column. This gives a complete initialization to the camera calibration and frustum reconstruction procedure.

Finally, we want to emphasize the importance of the initialization. There is a well-known (20) focal length and pyramid shrinkage ($f - \alpha$) ambiguity in an uncalibrated image, i.e. the same image can be explained as a pyramid ($\alpha < 1$) viewed by a camera with a certain focal length, or a prism ($\alpha = 1$) viewed by a camera with a different focal length. Hence, directly fitting a parameterized pyramid frustum and a camera model to the image cannot generate correct result. Either one has to be fixed in order to uniquely determine the other. Here, we provide such a technique for general uncalibrated images.

5.2.2 Symmetry-based Stereo

Many architectures exhibit symmetry. The two most common symmetries are bilateral symmetry and rotational symmetry. Both of them can be represented by the pyramid frustum as illustrated in Figure 5.5 (a) and (b) respectively. Bilateral symmetry is characterized by the symmetry plane, i.e. the x-z plane. (Some buildings exhibit further symmetry across the y-z plane.) Rotational symmetry is characterized by the rotation axis, i.e. the z-axis. Once the frustum shape, i.e. $\mathbf{\Lambda}$, is computed, the type of symmetry can then be automatically determined from the frustum shape.

From the calibrated camera, we can duplicate another virtual camera accord-

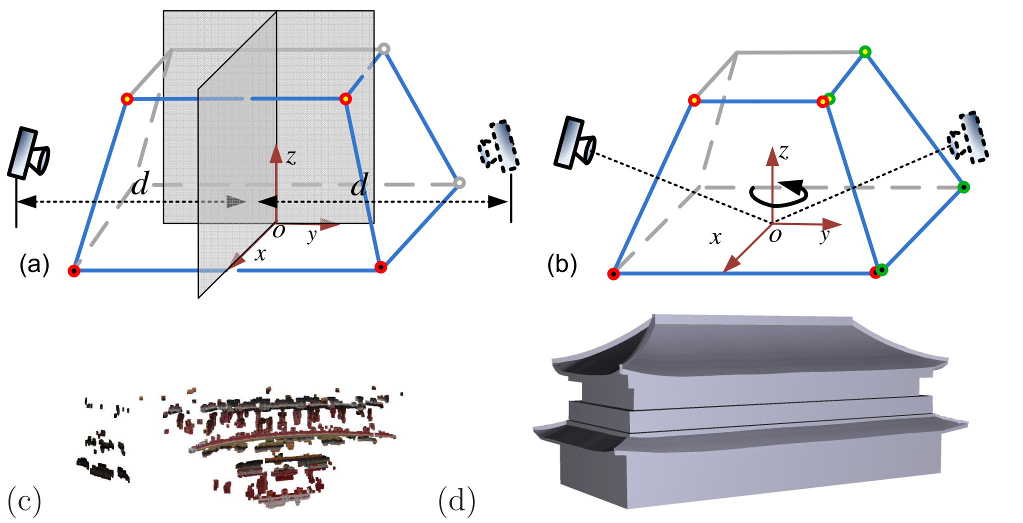


Figure 5.5: Representing architecture symmetry by pyramid frustum. (a) Bilateral symmetry is characterized by the symmetry plane, i.e. x-z plane. (b) Rotational symmetry is characterized by the rotation axis, i.e. z-axis. With the calibration of the real camera, a virtual camera can be duplicated according to the underlying symmetry. (c) Stereo algorithms can be applied to the real and virtual camera pair to recover a set of 3D points. (d) With a few strokes to delineate the key parts, the user can build an initial model from these 3D points.

ing to the underlying symmetry of the building. The virtual camera is generated by flipping the real camera across the x-z plane in the case of bilateral symmetry, or by rotating the real camera around the z-axis for an angle $\pi - \theta$ in the case of rotational symmetry. The observed image from the virtual camera can be derived from the input image. It is the input image with a horizontal flipping in the case of bilateral symmetry, or is exactly the input image in the case of rotational symmetry (33). Some previous works (21, 102) demonstrate that stereo algorithms can be applied to the real and virtual camera pair with manually specified correspondences.

As discussed in Chapter 1, feature detection and matching (53) can be performed automatically to establish correspondences between the real and virtual image pair. To facilitate matching, we take the frustum's side face as an initial estimation of the building façade, which induces a homography between the real and virtual image (30). We only consider matches consistent with this homography. To reduce projective distortion, the image region enclosed in the frustum side face is further 'rectified' by mapping it to a rectangle. Image features are computed in this 'rectified' rectangle. Obtained matches are further propagated according to the method described in (47). Then all matched features are reconstructed by triangulation (30), which generates a set of 3D points on the building façade. An example of this reconstruction is shown in Figure 5.5 (c). More examples of 3D reconstructions are included in Appendix C.

5.3 Surface Modeling

After the symmetry-based stereo, we obtain a set of 3D points on the building façade. Then the user can interactively build a surface model according to these points and the image information. The user first marks out large architectural components, such as walls and roofs, with a few strokes. The shapes of these components are automatically determined from the recovered 3D information. If further shape details are required, the user can also add roof tiles and handrails by a few additional strokes. The whole model is then textured according to the input image. In the input image, part of the model is imaged from a slanted view, which causes texture distortion. We use texture on fronto-parallel faces to correct this distortion. Textures on the invisible surfaces are synthesized by taking the weathering pattern into consideration.

5.3.1 Geometry modeling

Model initialization Multi-view stereo automatically reconstructs a set of 3D points. User interaction often follows to build surface model according to these points. This leveraging of automatic vision technique together with user interaction is employed in several previous systems, such as (81, 91, 96). Similarly, we interactively identify architectural components such as walls and roofs from the image plane, and then compute their 3D shape and position according to the reconstructed 3D information. Müller et al. (59) and Xiao et al. (96) propose an automatic method to partition building façades into rectangular components. However, these methods cannot handle complex façade data such as those shown in Figure 5.2 and Figure 5.12. The automatic partition of such complex façade

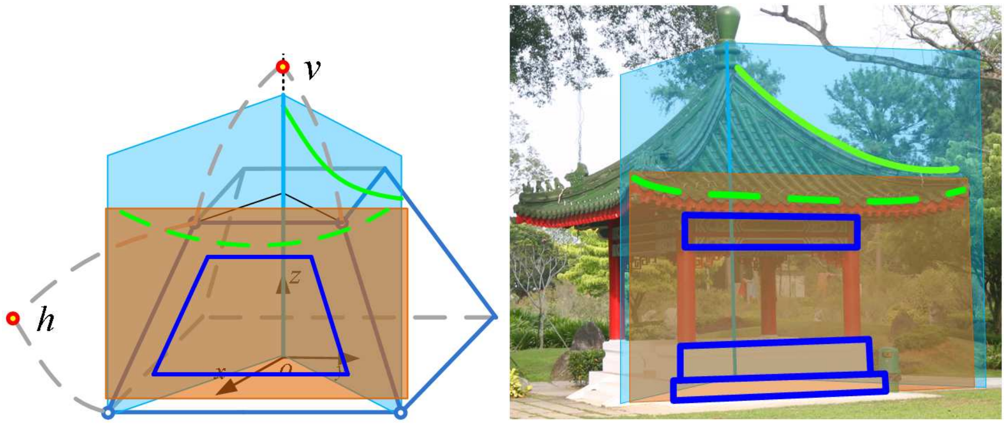
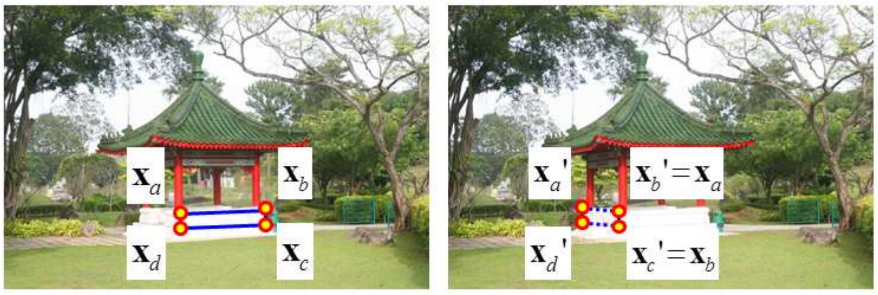


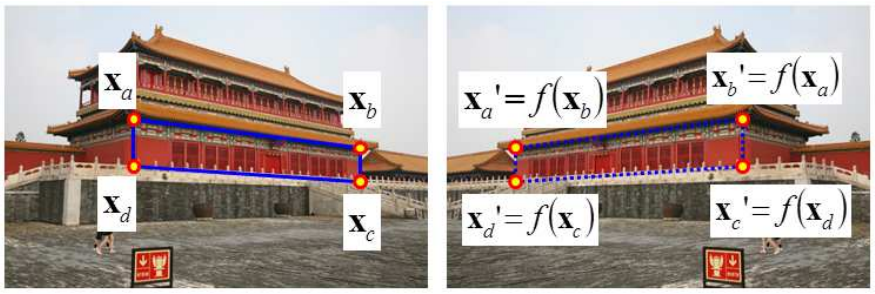
Figure 5.6: The user interactively draw a few strokes to build an initial model. The blue quadrilateral is the user-drawn wall structure. Green strokes are the user-drawn roof boundaries.

image is out of the scope of this thesis, and we rely on user interaction to mark out the architectural structures and leave the automatic partition for future research. We mark out two kinds of structures, namely planar walls and curved roofs.

The user interactively marks out a planar structure in the image. Then its position is determined according to the enclosed 3D points. Its symmetric counterparts are also automatically generated. Each planar structure should pass through two additional points, which are the intersection of the two pairs of edges of the frustum side face. We compare the image orientation of the user drawn strokes with that of the edges on the calibration frustum to determine which frustum side face to use as reference. In the left of Figure 5.6, these two points are illustrated as h and v. h is the vanishing point of the horizontal direction. v can be a finite or infinite point, depending on the parameter α of the frustum. Similar plane fitting approach is used in (81). These vanishing points serve the purpose of maintaining architecture shape regularities, such as the angle between two adjacent wall planes. Of course the user can also choose



(a)


$\mathbf{x}_a$
$\mathbf{x}_b$
$\mathbf{x}_d$
$\mathbf{x}_c$
$\mathbf{x}_a'=f(\mathbf{x}_b)$
$\mathbf{x}_b'=f(\mathbf{x}_a)$
$\mathbf{x}_d'=f(\mathbf{x}_c)$
$\mathbf{x}_c'=f(\mathbf{x}_d)$

(b)

Figure 5.7: Correspondences derived from intersection of wall edges. (a) In the case of rotational symmetry, the virtual correspondences for $\mathbf{x}_b$,$\mathbf{x}_c$ are given by $\mathbf{x}_a$, $\mathbf{x}_d$ respectively. (b)In the case of bilateral symmetry, the virtual correspondence for $\mathbf{x} = (x, y)^T$ is given by $f(\mathbf{x}) = (w - x, y)^T$, where w is the image width.

to discard these constraints to have larger modeling flexibility. In Figure 5.6, the blue quadrilateral is the user-drawn planar structure. With the two constraint points h, v, one reconstructed 3D point in the enclosed region can uniquely determine a planar structure. If multiple points are available, we apply RANSAC to obtain a robust fitting. If no reconstructed points are found in the enclosed region, the user is required to mark out the exact wall boundary (as shown in the right illustration in Figure 5.6) and the intersection of the four edges are used as corresponding points observed in the real and virtual images for triangulation. This is illustrated in Figure 5.7.

Generally, there are not enough reconstructed 3D points on the roof (see

Figure 5.5 (c)) to determine its shape and position. The user needs to mark out roof boundaries in 3D to model the roof. To simplify the user interaction, we introduce a set of auxiliary planes. The user draws 2D curves within these planes to decide 3D roof boundaries. Once the roof modeling function is enabled, multiple blue auxiliary planes are overlaid on the input image as illustrated in Figure 5.6.[2]

The user marks out the *back edge* of the roof, i.e. the solid green curve. Its 3D position is determined by projecting the drawn curve to the auxiliary plane according to the calibrated pinhole camera. All the symmetric counterparts of this *back edge* are generated according to the symmetry automatically. The roof hip, the beam along the *back edge*, is modeled by raising the 3D edge for a constant distance. Then the user draws the *front edge*, i.e. the dashed green curve in Figure 5.6. Similarly, its 3D position is obtained by projecting the drawn curve to the brown auxiliary plane, which is parallel to the z-axis and passing through the end points of the *back edge*. The curved roof is interpolated according to these surrounding edges. More details of the user interaction can be found in Appendix D. With these strokes to mark out walls and roofs, an initial 3D model of the building can be obtained as shown in Figure 5.5 (d). Corresponding strokes are shown in Figure 5.2 (a).

Model refinement Many architectures contain intricate geometric ornaments, which are hard to reconstruct from stereo triangulation. We describe an efficient way to model these details. We deal with two types of shape detail here, roof tiles and carved handrails. In addition, we also have predefined geometric primitives

[2]The auxiliary planes of bilaterally symmetric buildings are illustrated in Appendix D

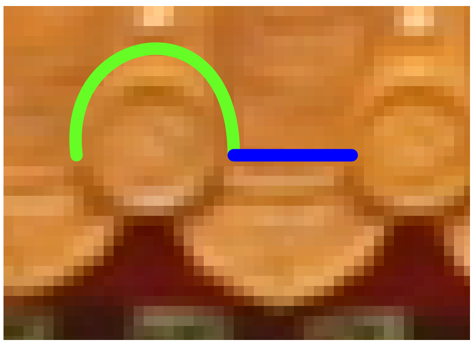
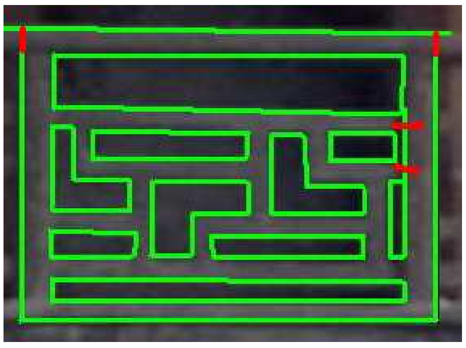
(a)  (b) 

Figure 5.8: Model refinement. (a) The user marks out the tile shape from a rectified view of the roof's front edge. This tile is applied to all roofs. (b) Cut pattern is automatically extracted. The user can also refine some incorrect line segments as illustrated in red.

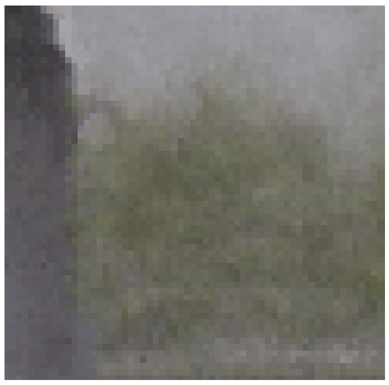
such as spheres, pillars and steps, which can be directly inserted into the building model. The user can also model revolved surface by specifying the revolving boundary.

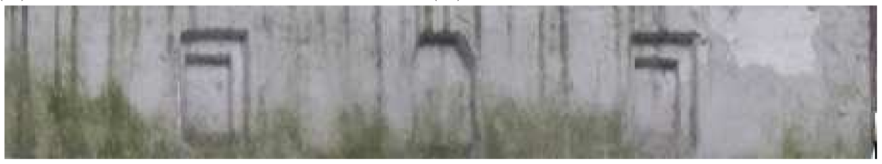
From the initial model, we extract a rectified view of the roof *front edge* as shown in Figure 5.8 (a). The user marks out one tile in this rectified view. The tile's shape and interval are determined according to these two strokes. The number of tiles is calculated by dividing the whole *front edge* length by the tile's size and interval. Then the tiling is applied to the whole roof surface. Each tile is textured by a predefined generic texture. Handrails have intricate cut patterns. We also extract a rectified view of the handrail as shown in Figure 5.8 (b). We apply Canny edge detection to extract edge pixels, which are then traced to form segments. We discard short segments and constrain remaining ones to have a few predefined directions, such as vertical or horizontal. In Figure 5.8 (b), the green edges are automatically detected, while the red strokes are user-input to refine incorrect edges. The user can either add or delete edges. The 3D shape of the handrail is created from the 2D pattern with a constant depth.

5.3.2 Texture Enhancement

One inevitable problem in single image modeling is the lack of texture samples. Some parts of the building are viewed from a slanted view angle or occluded in the input image. Texturing by back-projecting the image to the 3D model will cause large texture distortion. This distortion is systematically studied in (86), where the texture map is segmented and synthesized with consideration of orientation and scale changes. We also apply synthesis techniques to improve the texture quality with two novel features. First, we require the final texture to be consistent with the foreshortened image, which contains partial information of the underlying texture. Second, we require the synthesized texture to have consistent weathering patterns.

We enhance texture maps by applying patch based synthesis (43). We first generate an initial texture map by back projection. This texture map is marked automatically as regions free of distortion, with distortion, or occluded, by thresholding the ratio between the size of the mesh triangle and that of its image projection. Larger ratio indicates larger texture distortion. Texture in the distortion free regions is used as samples to enhance that of other regions. We treat the enhancement of foreshortened region as a 'super-resolution' problem (22). This enhancement runs by iterations. At each iteration, we overwrite a foreshortened texture patch by a distortion free patch that is most similar (in the sense of SSD) to it. This new patch is stitched to the texture map by a graph-cut optimization as in (43). A result of this super-resolution is shown in Figure 5.9. The texture in (a) has limited resolution due to foreshortening in the input image. Our method can enhance it to one with a similar resolution to the fronto-parallel view

(a) (b) 

(c) 

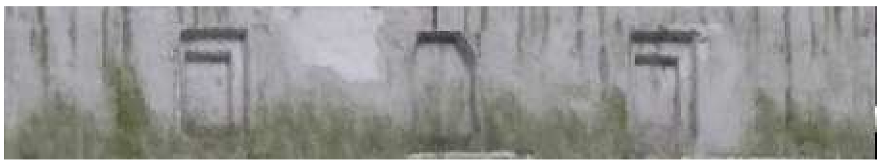
(d) 

Figure 5.9: Examples of texture enhancement. The texture in (a) has low resolution because of the foreshortening in the input image. This texture is enhanced to (b) to reduce artifacts. Texture (d) is synthesized according to the sample from texture (c). Similar weathering pattern is maintained.

(Figure 5.9(b)).

The simplest way to texture the occluded regions is to repeat the same texture as those of their symmetric counterparts. However, this simple repeating makes the model look artificial. Instead, we synthesize texture in the occluded regions according to those textures found in the distortion free region. Another difficulty lies in maintaining consistent weathering patterns. Architecture surfaces often have strong weathering patterns as shown in Figure 5.11. Wang et al. (92) propose to extract the surface appearance manifold and weathering degree map from image samples to generate physically correct result. Here, we seek a simple solution that yields plausible results. We observe that the weathering degree on buildings is generally inversely proportional to the height. Hence, we take the height as indicative of the weathering degree, which is used to guide the patch-based synthesis to generate the missing texture. We iteratively copy a patch from a distortion free region with consistent boundary and similar weathering degree to the occluded surface. The copied patch is also stitched with graph-cut optimization. To maintain semantic texture structures, such as doors and windows, we first copy them from the symmetric counterpart and keep them uncovered during synthesis. An example of this synthesis is shown in Figure 5.9 where (c) depicts the distortion free texture, and (d) depicts the texture synthesized according to (c).

5.4 Experiments and Discussion

5.4.1 Experiments

We first evaluate our symmetry based 3D reconstruction with a synthetic frustum image. We manually mark out 6 visible frustum vertices to reconstruct the frustum shape (Alternatively, we can also mark out the frustum edges and compute the 6 frustum vertices from the edge intersection automatically. This is easier for user interaction in the cases where frustum vertices cannot be identified accurately, e.g.Figure 5.13). In our experiments, the user clicks often deviate from the true vertex position by 1.3 pixels (in an image of resolution of 1200x800). The error of focal length is 4.3% of the true value, and the mean error of vertex position is 0.1% of the distance between the camera and pyramid frustum center. The reconstructed frustum (shown in red wireframe) is verified from a novel

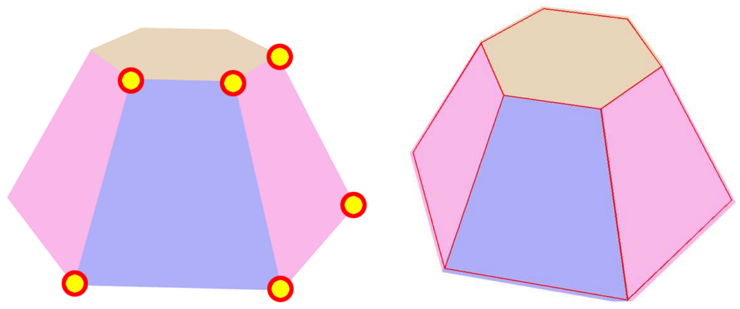
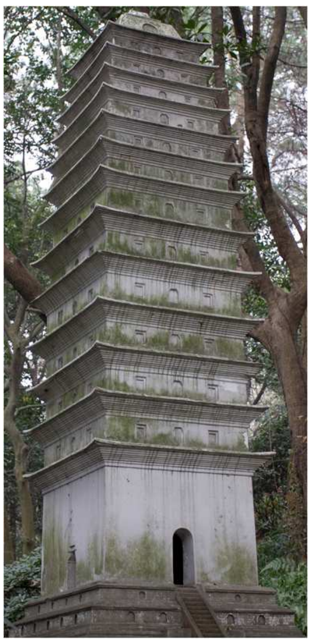
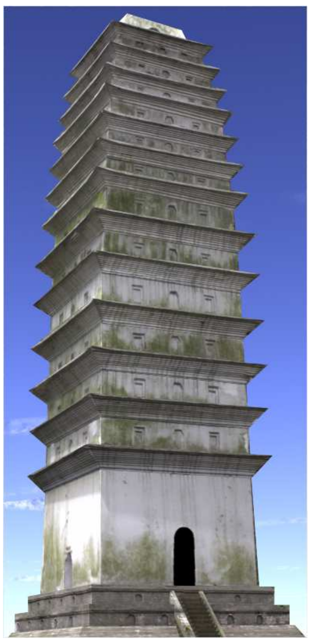
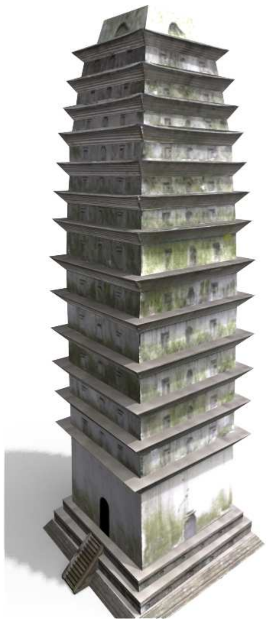


Figure 5.10: Validation of symmetry based reconstruction on synthetic data. On the left is the input image (overlaid with user clicked 6 vertices). On the right is a rendering from a novel viewpoint. The reconstructed shape is overlaid on the image (drawn as red wireframe).

viewpoint as shown in the right of Figure 5.10. Then we verified the accuracy of estimated focal length in real photo of Figure 5.1 (b). The true focal length calculated from photos of a checkerboard is 2864.8 pixels (50 mm). In 10 trials, our estimated focal length varies from 2582.4 pixels to 3069.8 pixels. In comparison, we also implemented the calibration method discussed in (15) which generates results varying from 2029.4 pixels to 3968.8 pixels.

We test our method on several examples with different level of complexity. Our symmetry-based triangulation takes 4-5 minutes on a PC with 2.83GHz CPU and 4GB memory. With the recovered 3D points, the user draws strokes to build an initial model (strokes for each example are provided in Appendix D). We measure the user effort by the interaction time, because the other parts are automatic. It takes $2 - 10$ minutes user interaction to generate a result. We report the user interaction time and the number of reconstructed 3D points for each example in the Table 5.1.[3]

[3]This interaction time is measured for an experienced user. We also let a novice try our system. After watching a 10 minutes instruction of the system (with the pavilion example), he spends 9 minutes in total to model the Berkeley Campanile (5 minutes for the automatic triangulation and 4 minutes for user interaction; no texture synthesis is applied for this example).

(a)  (b)  (c) 

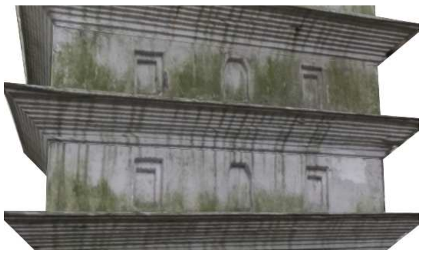
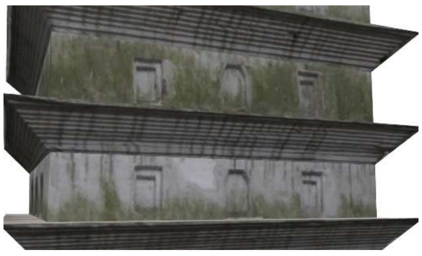
(d)  (e) 

Figure 5.11: A pagoda example. (a) is the single input image. (b) is the recovered model rendered from the same viewpoint as the input image. (c) is the rendering from a novel viewpoint. (d) and (e) are two different façades at the same height. The façade in (d) is textured from the input image; the texture in (e) is synthesized by our method. Our texture synthesis generates more vivid texture than simple repetition.

As no ground truth 3D model is available, here we only evaluate the results visually against the input image. Figure 5.2 shows an example with bilateral symmetry (the TiRen Ge in the Forbidden City). Figure 5.2 (a) shows the user-drawn strokes. The highlighted six points are as the corners of the pyramid frustum. Feature matching is most effective on this example, with 12,000 3D points recovered as shown in Figure 5.5 (c). Figure 5.2 (b) shows the rendering of the recovered model from the same viewpoint as the input image. A novel viewpoint rendering is provided in Figure 5.2 (c) for validation. A simple pavilion example is shown in Figure 5.1 (b), where the left image is the input image and the right image shows the model rendered from the same viewpoint for validation. A ridge on the top is missing, which is caused by the inaccuracy in the interactive modeling. The roof hip is a little higher than it should be, and thus occludes part of the second ridge. Figure 5.11 shows an example with multiple floors. We first model the first floor and apply the modeled result to the other floors with only one stroke to compute a scaling and vertical translation (please refer to the supplementary video). This example highlights our texture enhancements. Figure 5.11 (d) and (e) show two different building façades at the same height; while the façade in (d) is textured according to the input image, the texture in (e) is synthesized using our method. Our synthesized texture produces consistent weathering pattern. Figure 5.12 shows a complex pagoda with rotational symmetry. It has highly curved roofs, which are different at each floor. We draw 3 strokes to model each roof (1 additional stroke for the *back edge* to model the shrinking of the hip). Furthermore, its handrail has intricate cut patterns. It takes about 10 minutes of user interaction to model this finely detailed example. Figure 5.12 (d) shows a close-up view. Further geometrical details are highlighted by the wireframe rendering in Figure 5.12 (e).

Our method is also applicable to the modeling of western buildings, as the various principles invoked in this chapter are generally true for most architectural forms. The Eiffel Tower in Figure 5.13 and the Berkeley Campanile in Figure 5.14 show two western buildings modeled using our method. We model the curved surface of the Eiffel Tower according to its curved silhouettes. One of the cut patterns is modeled in the same way as the handrail. The Berkeley is the easiest as there is no curved roof and cut patterns. It is modeled with less than 2 minutes interaction.

5.4.2 Discussion

Architecture modeling has been an active research field for many years. Existing systems, such as (16, 81, 96), create highly realistic results from multiple images. In comparison, we seek to provide an alternative solution for architecture modeling when only a single image is available. To achieve this, a novel method is designed to calibrate the camera and recover 3D scene points from a single image. The recovered 3D information helps to reduce the amount of user interaction by avoiding tedious manual correspondence. We also enhance the texture quality to improve single view modeling. Our method does not require complicated geometric and photometric image alignments. It can be a standalone toolkit for artists to create 3D architecture models from online pictures or archive pictures; or integrated into previous multi-view based systems to provide further shape constraints.

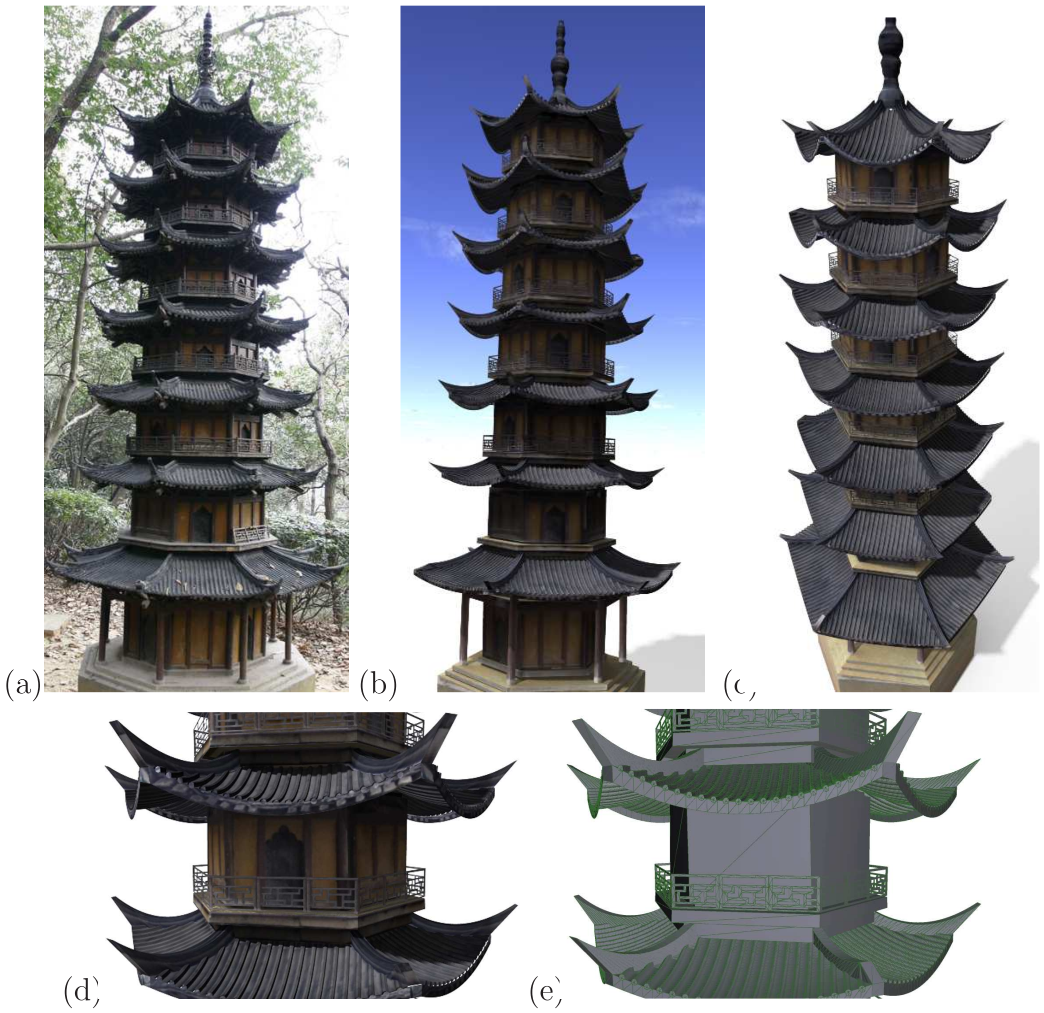


Figure 5.12: A pagoda with highly curved roof. Each roof is different from the others. (a) is the single input image. (b) is the recovered model rendered from the same view as the input image. (c) is the rendering from a novel viewpoint. (d) shows a close-up view of the building. (e) is the shaded wireframe to highlight the geometry. (Please zoom in the electronic version.)

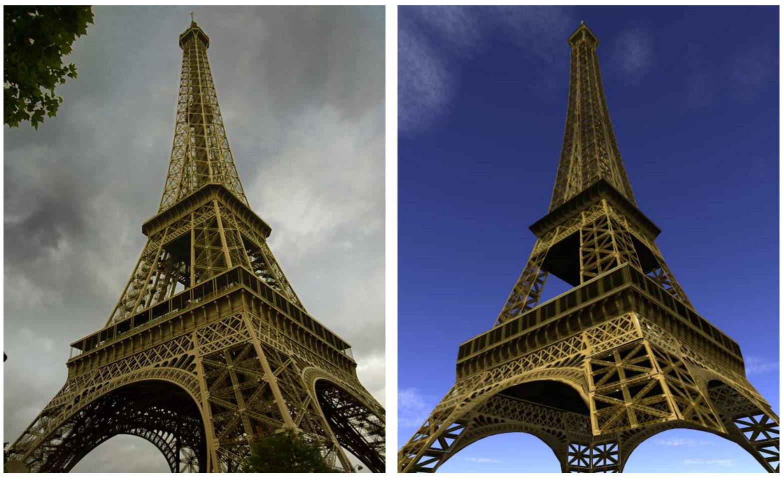


Figure 5.13: The Eiffel Tower example. On the left is the single input image. On the right is the recovered model rendered from the same viewpoint as the input image.

Limitations The current method has several limitations. First, like most image based modeling systems, our method prefers the input images to be free of shadow effect. Otherwise, shadow could be mistaken as texture and cause artifacts in the rendered models. Second, in the case that no symmetry is present (though a rare case), we cannot model the building from a single image. If the building can be decomposed into multiple symmetric parts, we might still model it part by part. If multiple images are available, our interactive system for decomposing and modeling architecture components can still be applied. However, we cannot reduce the interactions by symmetry. In that case, our method will be a regular multi-view interactive modeling system like (81). Third, our camera calibration relies on the quadratic initialization, which requires the principle point to be close to the image center unless both the length ratios and the frustum angle in Equation (5.5) are known. Last, large lens distortions could be a problem for our current single-view camera calibration algorithm.

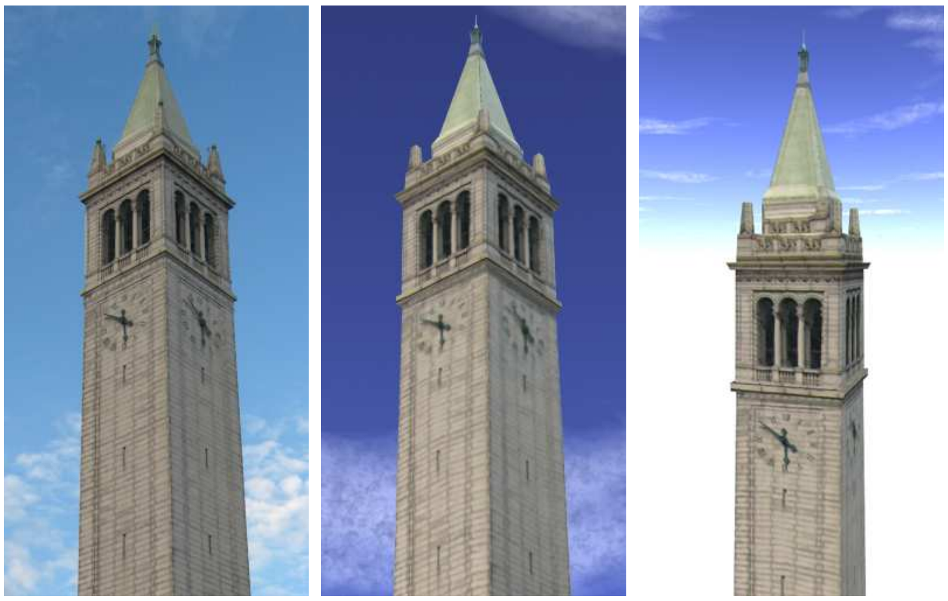


Figure 5.14: The Berkeley Campanile example. From left to right, they are: the single input image, the rendering from the same viewpoint as the input image and the rendering from a novel viewpoint.

Examples:	TiRen Ge	pavilion	P1	P2	ET	BC
# 3D pts ($\times 10^3$):	12	5	7	11	0.5	0.7
IT: (mins)	5	2	2	10	5	2

Table 5.1: Modeling statistics. We show the number of reconstructed 3D points and the user interaction time for each example in this chapter. Most of our examples require less than 5 minutes of user interaction. Note: IT = user interaction time, P1 = the pagoda in Figure 5.11, P2 = the pagoda in Figure 5.12, ET = the Eiffel Tower, BC = the Berkeley Campanile.

There are several ways to improve our current system. For example, in the current system, the roof tile is manually marked out from the rectified view. This part can be automated by image analysis and shape template matching. Image processing techniques can also be applied to snap user strokes to image edges to make the interface more efficient.

Chapter 6

Conclusion

In this thesis, we studied the paradox relationship between symmetric structure and image-based modeling.

3D reconstruction from unordered image collection could be ambiguous when there are duplicate or similar structures in the scene. We proposed to analyze the 3D reconstruction with respect to image content in a holistic fashion. First, we defined a new appearance based objective function for evaluating the optimality of a 3D reconstruction. We proved that this function always retain its global minimum for the correct 3D reconstruction when the appearance similarity evaluation error is within tolerance given in Appendix A. The optimization of this objective function, however, is a challenging problem. We designed an efficient algorithm to search for the optimal 3D reconstruction among the space of spanning trees defined on the image match graph. We used three strategies to speed up the searching process. Firstly, with careful analysis of the motion consistency between epipolar geometries, we narrowed down the search space to the number of inherent ambiguous solutions arising from mismatches. Secondly, intermediate reconstruction can be re-used and only local transformation is required to obtain the global camera poses in each iteration. Lastly, visited 3D reconstruction can be cached as a binary array representing its associated match graph. Experimental results showed that our technique outperforms the state-of-the-art algorithms on highly ambiguous scenes.

Knowledge of symmetry and regularity is useful for subsequent image-based 3D modeling, especially for modeling architectures. Automatic symmetry detection with existing 2D and 3D symmetry analysis on these type of data, i.e., images with strong perspective distortion and occlusion or sparse 3D point clouds, is difficult. Hence, we jointly analyzed the multiple input images and the 3D point clouds to robustly detect repetitive structures in the recovered scene. Image feature descriptors associated with reconstructed 3D points are used to help identify 3D points at symmetric positions. These symmetric 3D points are then analyzed to hypothesize symmetric relationships within the recovered 3D structure. This joint analysis can handle very challenging data that could fail most conventional symmetry detection algorithms and bridges the gap between pure image based 2D symmetry analysis and point cloud based 3D symmetry analysis.

Creating 3D architecture models from sparse point clouds is challenging. Most existing methods make assumption about surface planarity and cannot model architectures with curved surfaces and intricate details well. Interactive modeling of architectures with complex geometry is often time consuming. We exploited two types of symmetry, namely rotational symmetry and bilateral symmetry that are most commonly observed in architectures for interactive 3D modeling. We proposed a novel calibration method based on symmetry with a single image, and designed an interactive modeling system that can model architecture with complex geometry and intricate details efficiently. Symmetry property was also utilized for texture enhancement with a single input image. Therefore, we were able to create a highly photorealistic architecture model within minutes of user interaction.

What we have studied are difficult problems in computer vision and image-based modeling. There are still many issues unaddressed in the limited scope of this thesis. One of them is the robust 3D reconstruction of city-scale image collections. The state-of-the-art reconstruction system has already achieved significant improvement on the speed of 3D reconstruction computation (1), but the problem of selecting suitable images and correct image matches is still an open issue. In order to extend our holistic analysis of the optimality of a reconstruction to such large-scale data, careful analysis of the match graph is required for searching the solution space more efficiently. For instance, epipolar geometries which are consistent with any global camera configuration have no association ambiguity and should be excluded from the search space. Ambiguities only exists among epipolar geometries that are classified as inliers or outlier in different global camera configurations.

Another important issue is that the way the partial reconstructions obtained from view pairs getting registered and merged is still somewhat ad-hoc for most well-known systems. In the incremental approach, the number of images that should be added at a time and the frequency the bundle adjustment should be performed is adjusted accordingly in different systems. Some systems are optimized for speed, and some are optimized for robustness and accuracy. A global solution that can solve for initial camera poses and point locations on general data efficiently and robustly is still missing. Crandall et al. (13) formulated this problem as a global solution to a MRF problem, which, however, requires GPS information and near-planar camera motion.

Symmetry information is helpful for efficient modeling, and could also be helpful for 3D reconstruction (10). Interestingly, these 3D symmetries can be identified from image mismatches in the match graph. The problem here is how to reliably discover such regularities from suboptimal 3D reconstruction and be utilized to improve the reconstruction accuracy.

Dense reconstruction (i.e. dense point clouds) is usually required for automatic modeling system. While camera geometry and sparse reconstruction has been well studied, the dense correspondence problem still has many open issues, e.g. the handling of textureless regions, non-lambertion surfaces, large view point and lighting changes, etc.

Given a huge amount of unstructured 3D points, it is still a challenging problem to create high quality 3D models. Semantic understanding of the images and segmentation in both 2D and 3D space are required to identify points belonging to trees, ground, buildings and other objects, so that object specific prior and modeling techniques could apply. This approach has been successfully demonstrated for urban street-view images in (97), where the scene is densely dominated by buildings and there is nearly no disturbance from moving vehicles and pedestrains. For cities with heavy occlusion of greenery on the street level, satellite images are useful for semantic segmentation of the point clouds. Therefore, the registration of street-view images and satellite images, or in general, ground level images and 2D ground maps are necessary to achieve robust semantic analysis. Such solution will be useful for both outdoor and indoor applications on reconstruction, modeling, and navigation in the virtual and the real world.

I shall conclude the thesis with the statement that 3D reconstruction and modeling are important computer vision and computer graphics problems, and the study of higher level object properties such as symmetry benefits both reconstruction and modeling process. With the maturing of such technologies, different pieces of low-level and high-level visions are getting integrated into intelligent vision systems, which changes the way we interact with and live in the physical world.

Appendix A

Proof of Global Minimum

The normalization term $\frac{1}{\sum_{i=1}^{N} M_i}$ in Equation 3.9 remains unchanged for different 3D reconstructions. Given a 3D reconstruction, the objective function evaluated for feature points in image i is

$$\begin{aligned}
&\sum_{p=1}^{M_i} \hat{P}_{missing}(p) = \frac{1}{N}\sum_{p=1}^{M_i}\sum_{j=1}^{N} P_{missing}(p,j) \\
= \; &\frac{1}{N}\sum_{j=1}^{N}\{\sum_{p\in\mathcal{S}_{ij}^{++}} P_{missing}(p,j) + \sum_{p\in\mathcal{S}_{ij}^{--}} P_{missing}(p,j) \\
+ \; &\sum_{p\in\mathcal{S}_{ij}^{-+}} P_{missing}(p,j) + \sum_{p\in\mathcal{S}_{ij}^{+-}} P_{missing}(p,j)\}.
\end{aligned}$$

Here, we exchange the sequence of the two summation and partition the feature points in image i into four sets, $\mathcal{S}_{ij}^{++}, \mathcal{S}_{ij}^{--}, \mathcal{S}_{ij}^{-+}$ and $\mathcal{S}_{ij}^{+-}$ according to their evaluation in each image j. The first plus/minus sign denotes the feature point is actual visible/invisible in the image j. The second plus/minus indicates it is detected as matched/missing according to our feature matching criteria. Different

3D reconstructions give different partition of the image features. In the ground truth reconstruction, points in $\mathcal{S}_{ij}^{+-}$ (or $\mathcal{S}_{ij}^{-+}$) should go to $\mathcal{S}_{ij}^{++}$ (or $\mathcal{S}_{ij}^{--}$). Both $\mathcal{S}_{ij}^{+-}$ and $\mathcal{S}_{ij}^{-+}$ become empty. Hence, moving from any 3D reconstruction to the ground truth, the change in the objective function (evaluated in the image i) is

$$\begin{aligned}\Delta_i &= \frac{1}{N}\sum_{j=1}^{N}\{\sum_{p\in\mathcal{S}_{ij}^{-+}}(\alpha - P_{missing}(p,j)) \\ &+ \sum_{p\in\mathcal{S}_{ij}^{+-}}\left(P'_{missing}(p,j)-\alpha\right)\}.\end{aligned}$$

where $P'_{missing}(p,j)$ is evaluated with the ground truth reconstruction. Therefore, the inequality $\Delta_i < 0$ will hold, as long as

$$\tilde{P'}^{+-}_{missing} < \alpha < \tilde{P}^{-+}_{missing}.$$

Here, $\tilde{P'}^{+-}_{missing}$ and $\tilde{P}^{-+}_{missing}$ are the average of $P'_{missing}(p,j)$ and $P_{missing}(p,j)$ over all images and over the two sets $\mathcal{S}_{ij}^{+-}$ and $\mathcal{S}_{ij}^{-+}$ respectively. Typically, $\tilde{P'}^{+-}_{missing}$ is close to 0, while $\tilde{P}^{-+}_{missing}$ is close to the average percent of non-repetitive 'background points' in the image i. Hence, with an appropriate α the global minimum of Equation (3.9) is associated with the ground truth. However, when there is no non-repetitive 'background points' (i.e. $\tilde{P}^{-+}_{missing} = 0$, as the case in Figure 3.8 (b)), no suitable α can be found and our method will fail.

Appendix B

Lattice Detection Comparison

We report the detailed comparion of repetitive structure detection results with (67) here. The following tables give comparison between two methods. Successful repetitive structure detection is indicated by blue box; and red box otherwise. For each column pair, we list the lattice detection results obtained using (67) on the left, and the one obtained using our joint analysis on the right.

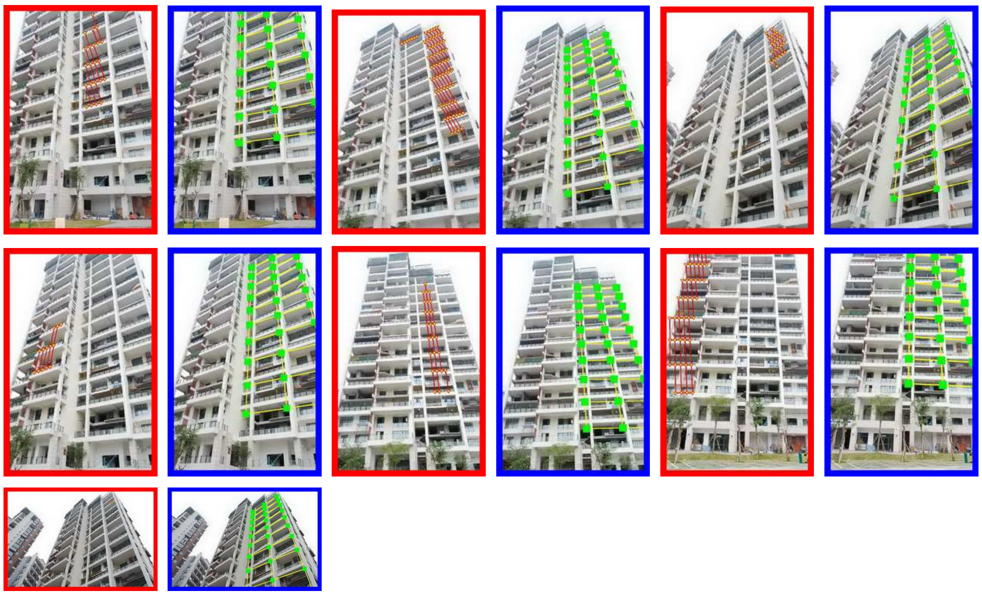


Table B.1: Lattice detection comparison with (67) on data 1.

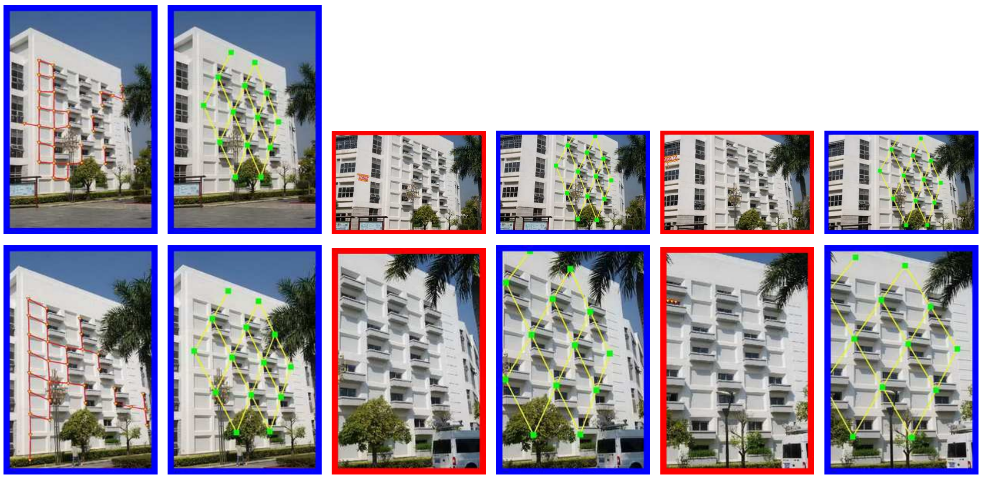


Table B.2: Lattice detection comparison with (67) on data 5.

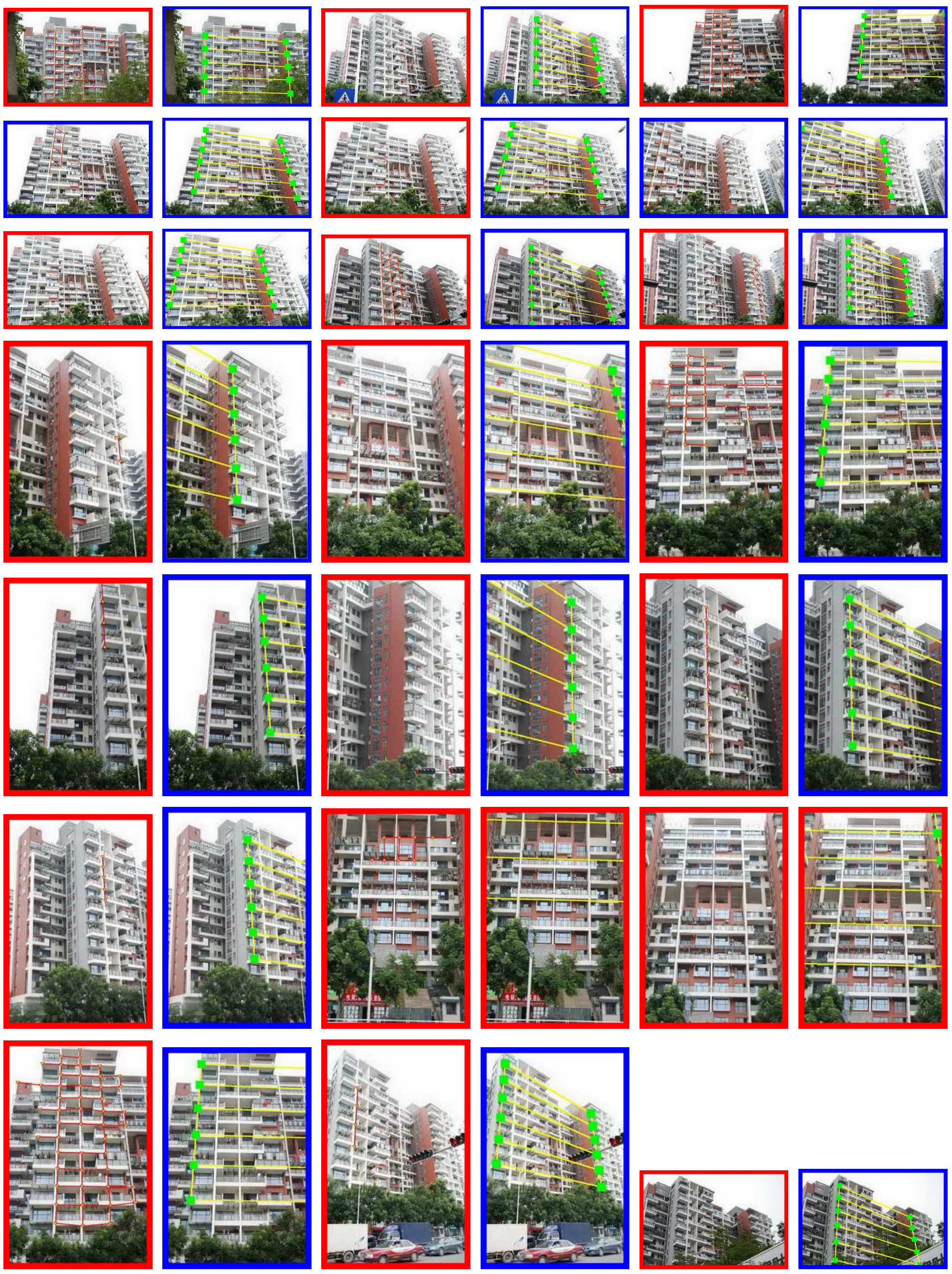


Table B.3: Lattice detection comparison with (67) on data 2.

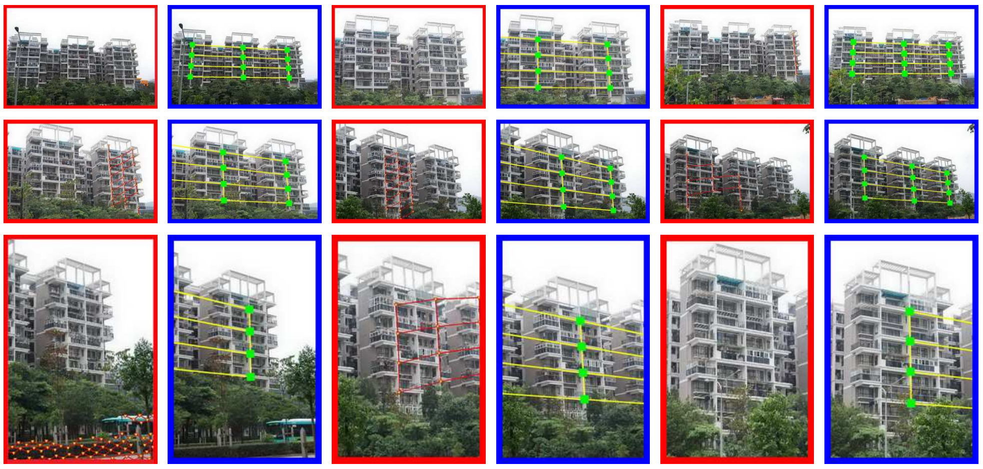


Table B.4: Lattice detection comparison with (67) on data 3.

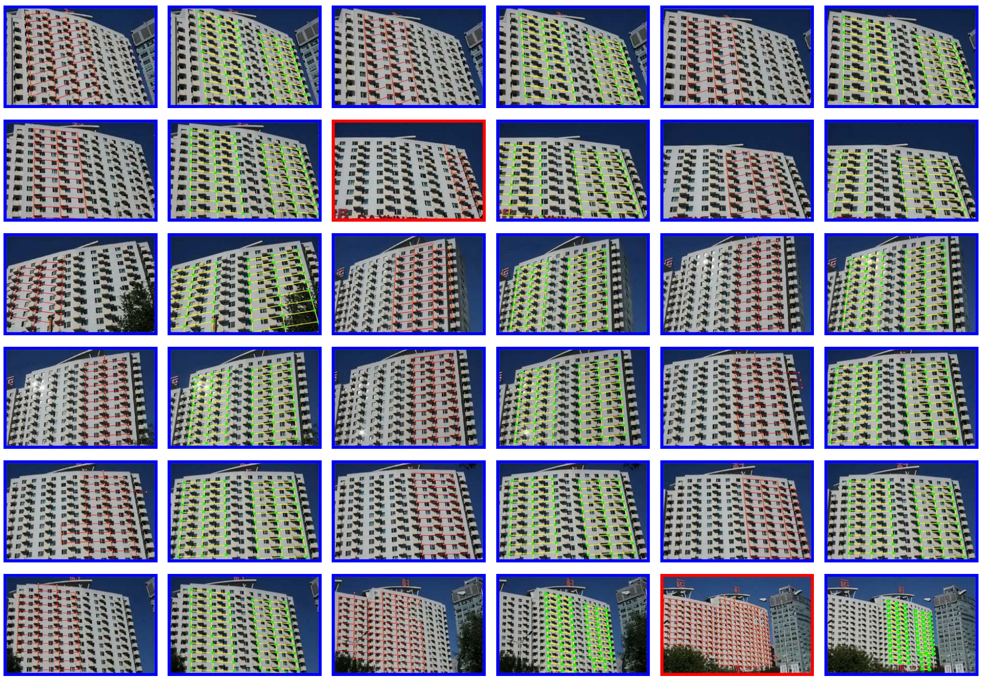


Table B.5: Lattice detection comparison with (67) on data 4.

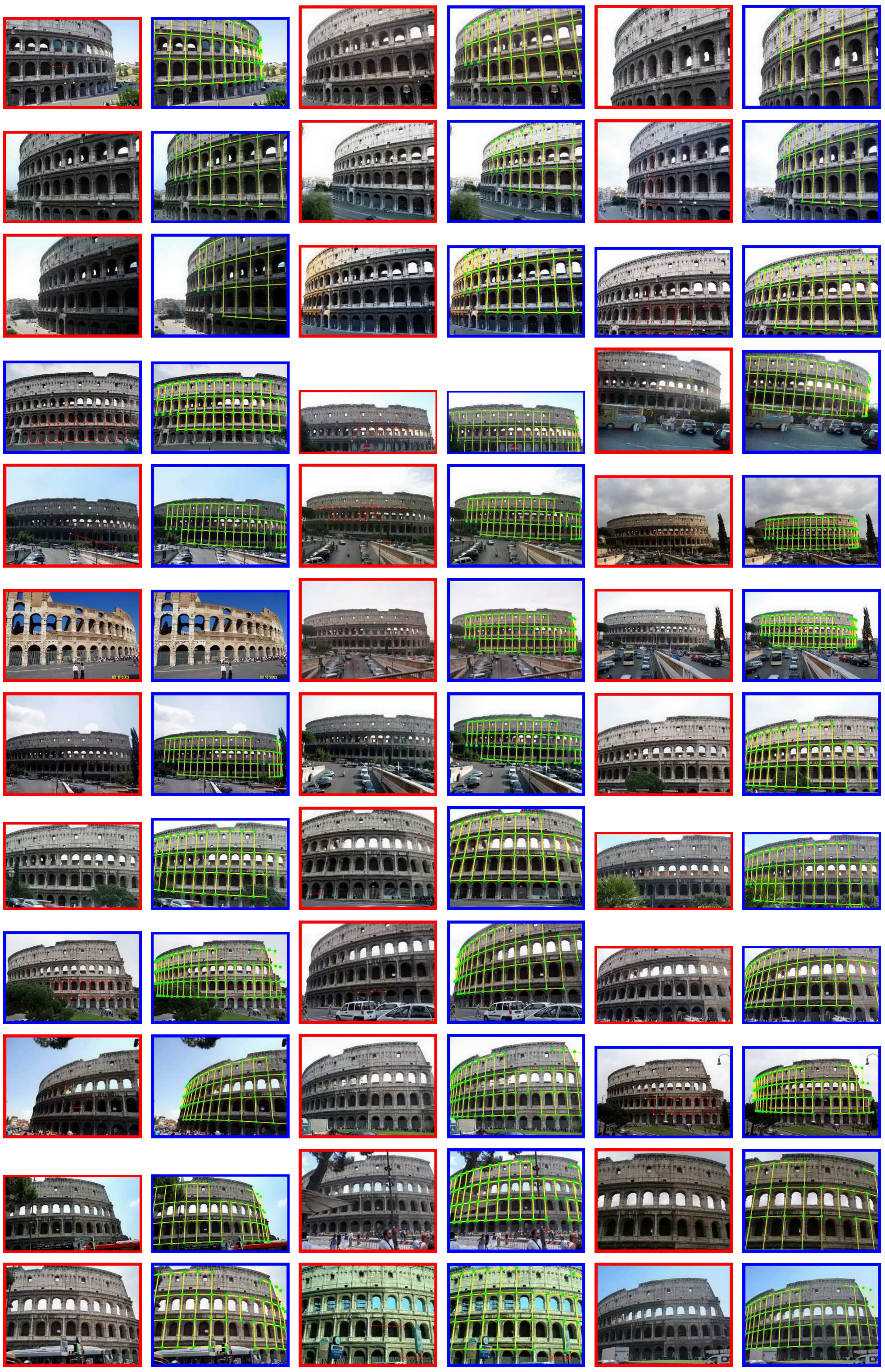


Table B.6: Lattice detection comparison with (67) on data 6 (continued on the next page).

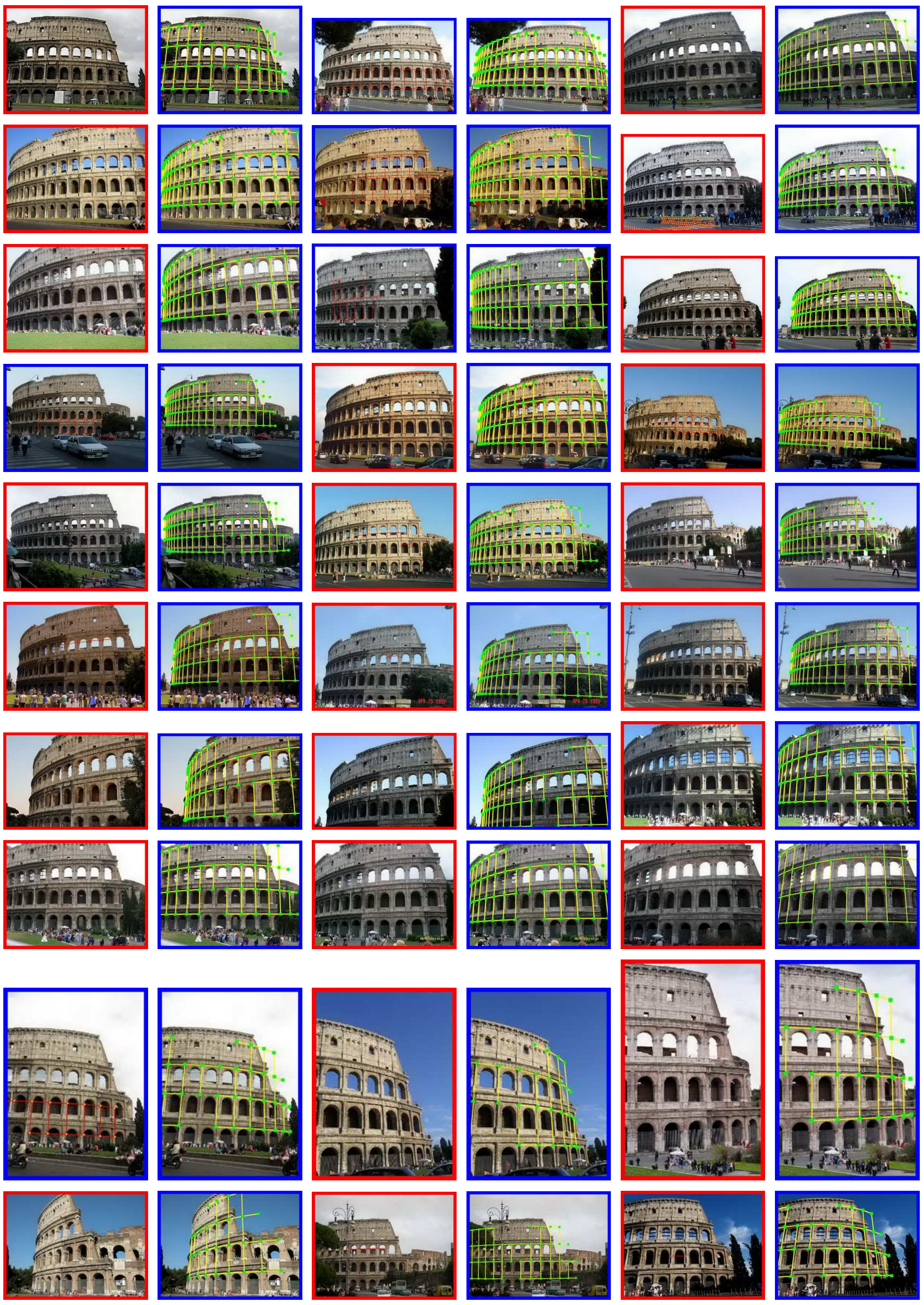


Table B.7: Lattice detection comparison with (67) on data 6 (continued from previous page and continued on the next page).

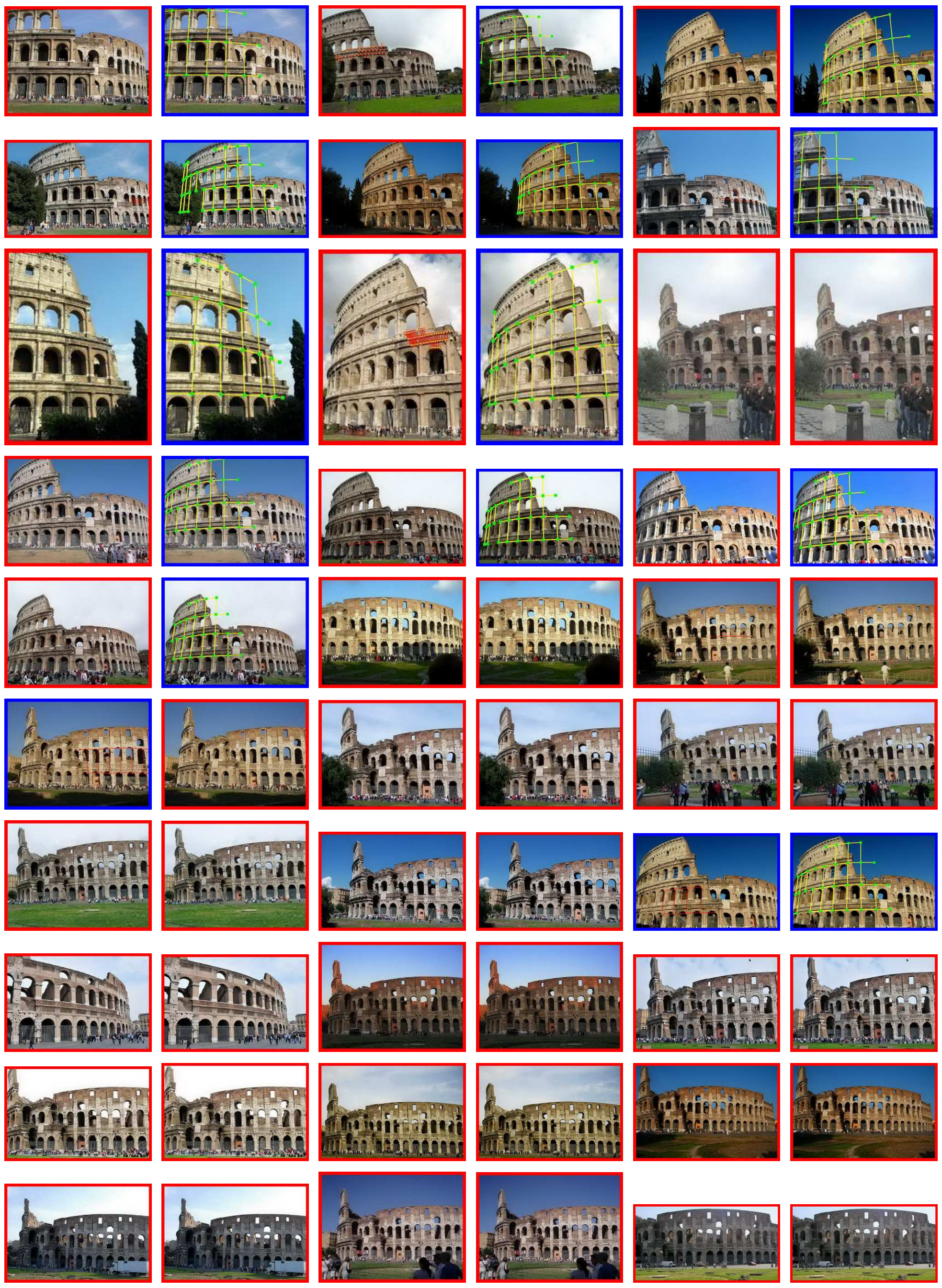


Table B.8: Lattice detection comparison with (67) on data 6 (continued from previous page and continued on the next page).

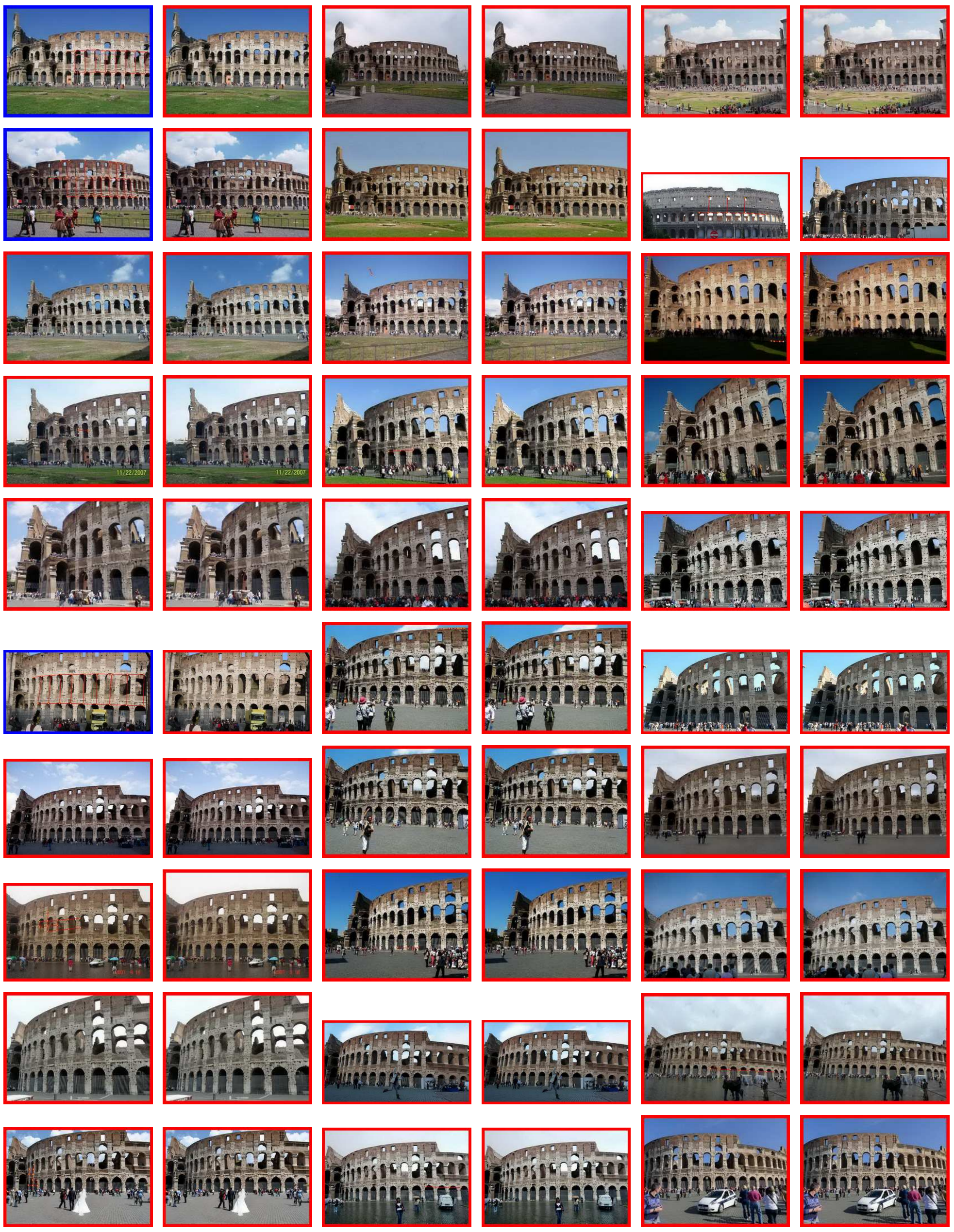


Table B.9: Lattice detection comparison with (67) on data 6 (continued from previous page and continued on the next page).

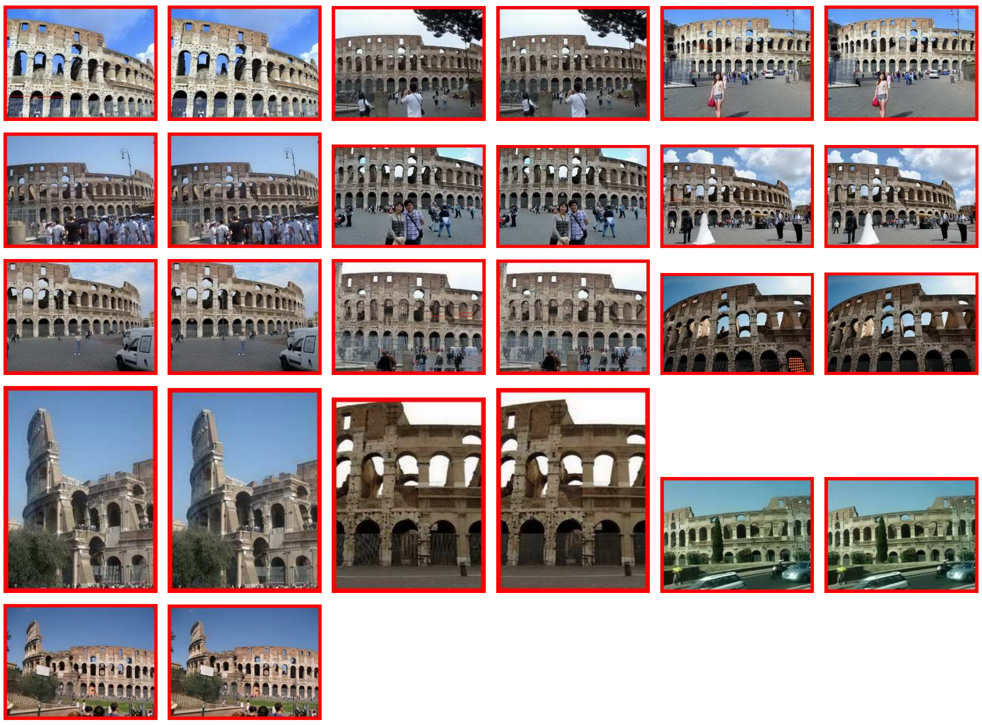


Table B.10: Lattice detection comparison with (67) on data 6 (continued from previous page and continued on the next page).

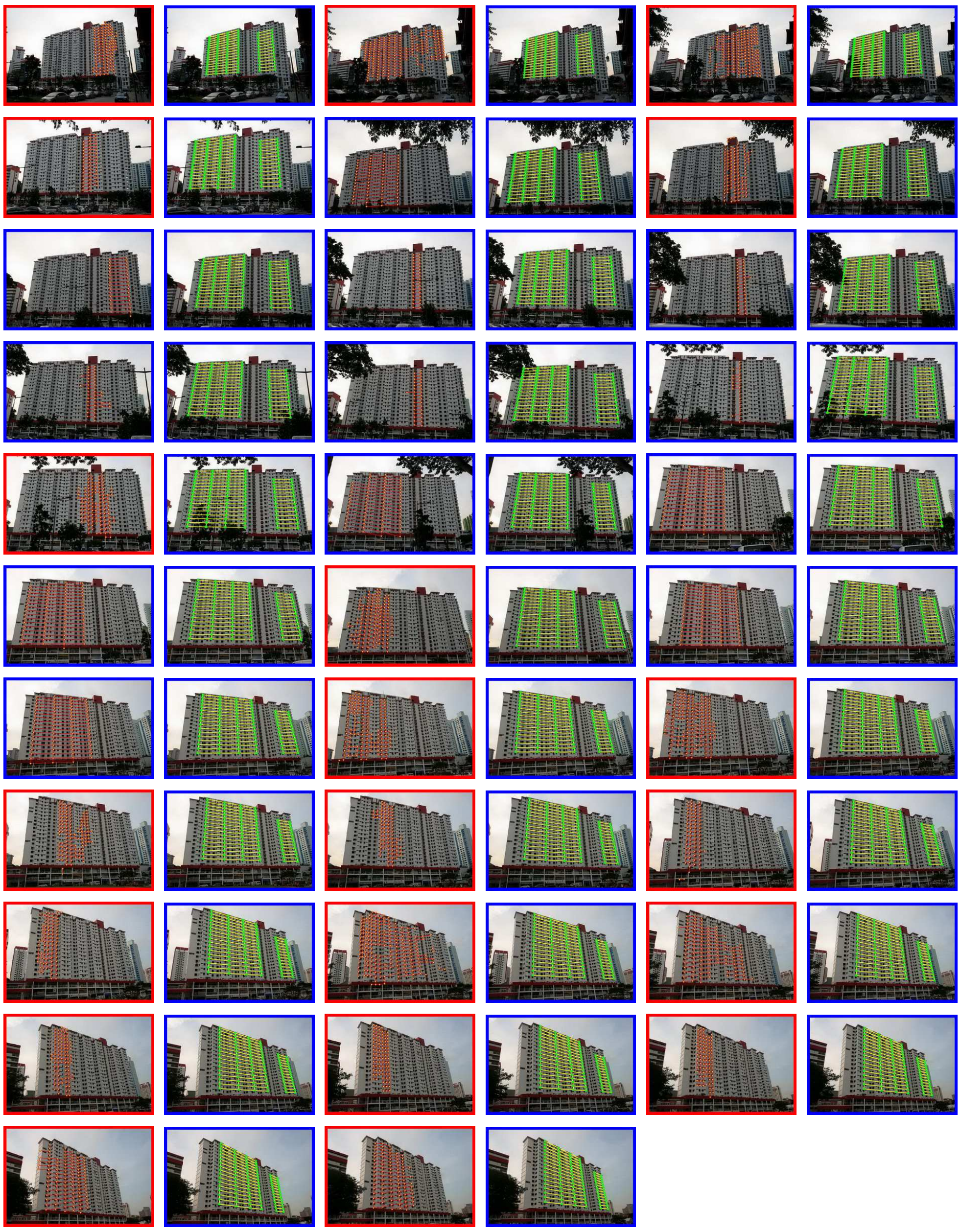


Table B.11: Lattice detection comparison with (67) on data 7.

Table B.12: Lattice detection comparison with (67) on data 8.

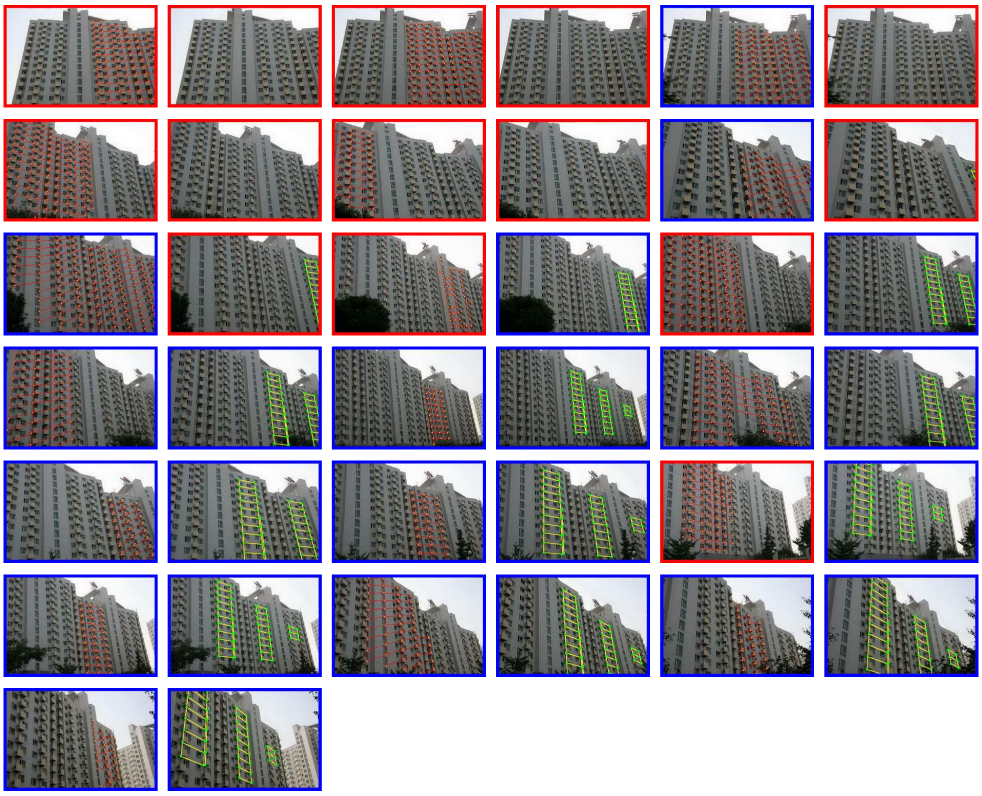


Table B.13: Lattice detection comparison with (67) on data 13.

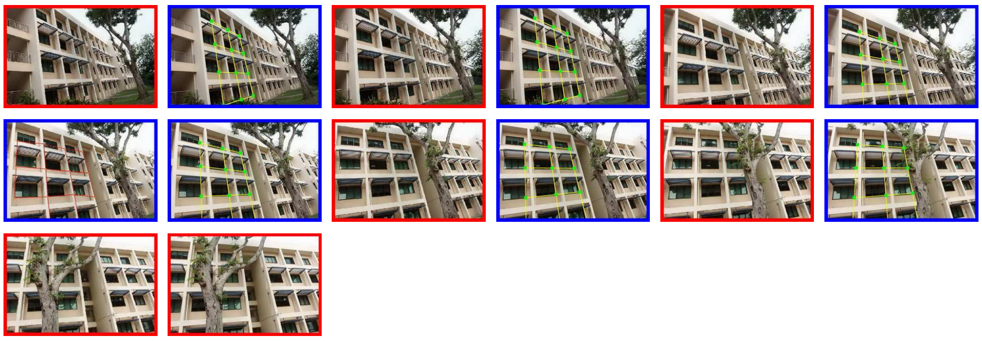


Table B.14: Lattice detection comparison with (67) on data 9.

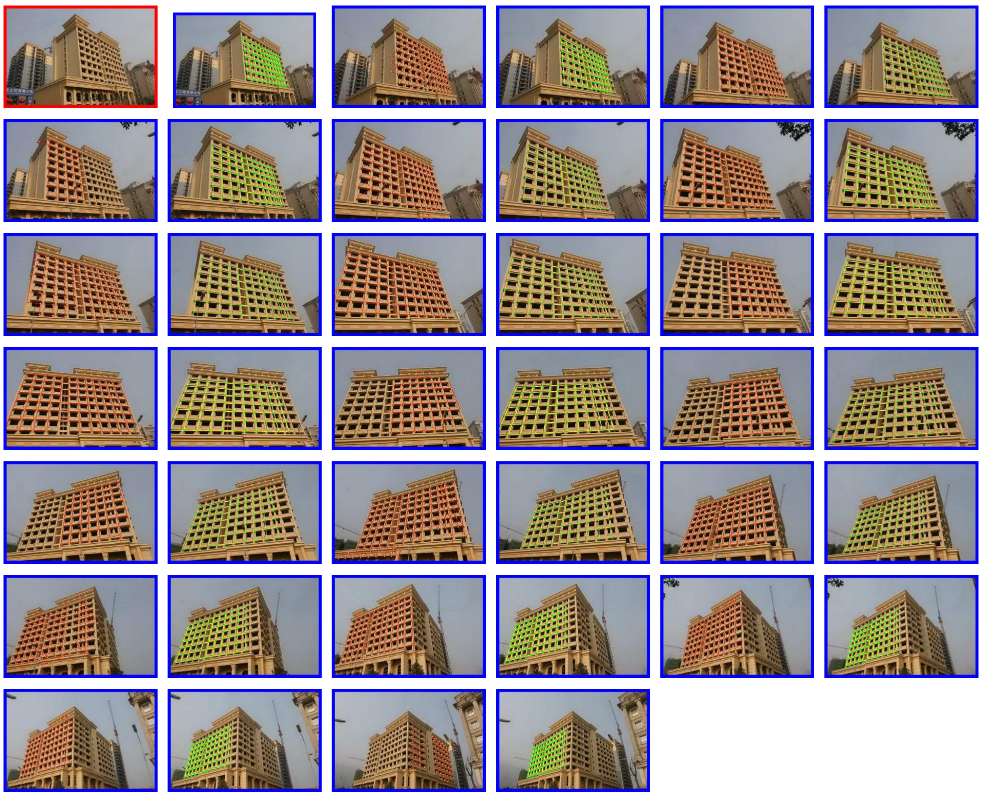


Table B.15: Lattice detection comparison with (67) on data 15.

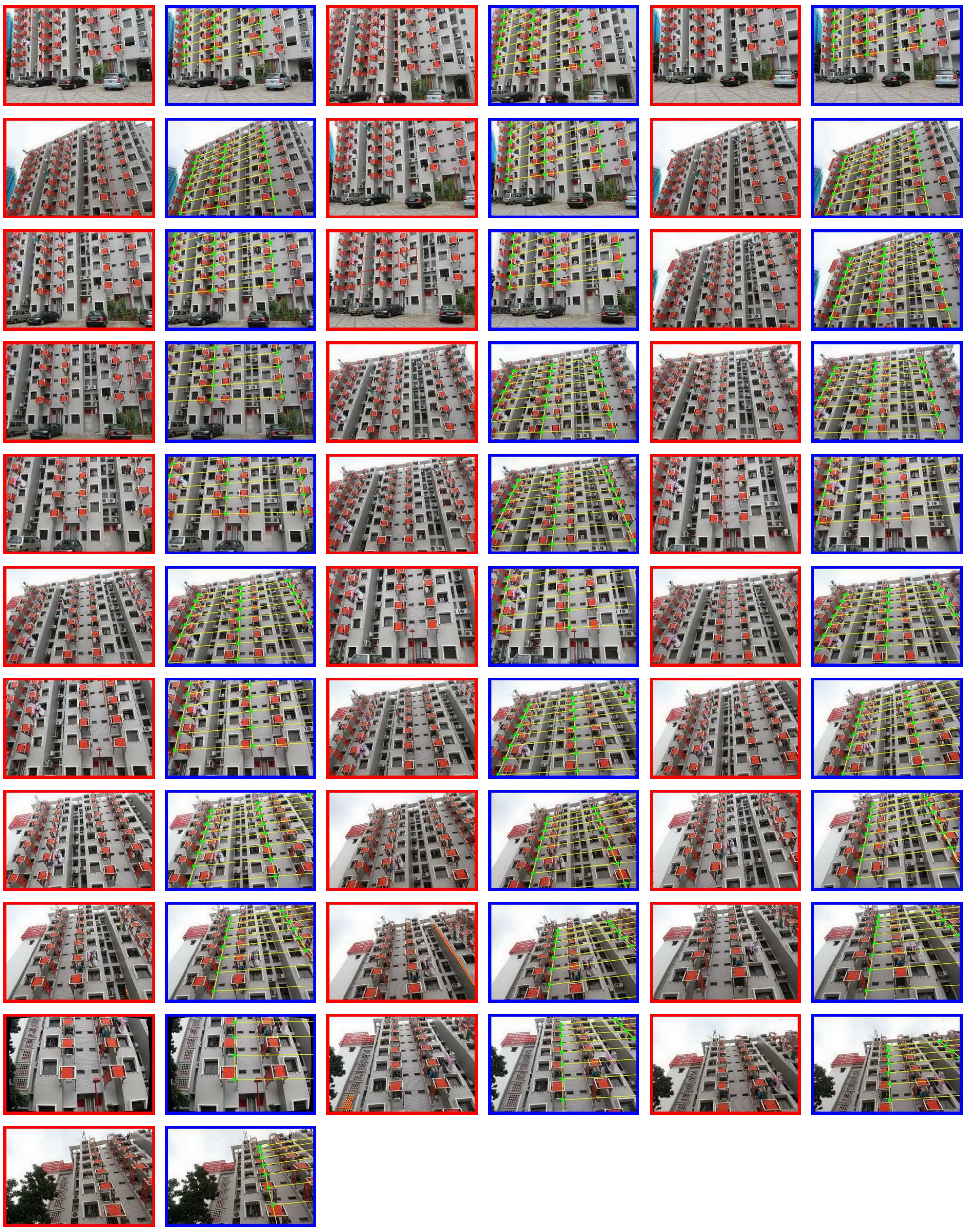


Table B.16: Lattice detection comparison with (67) on data 10.

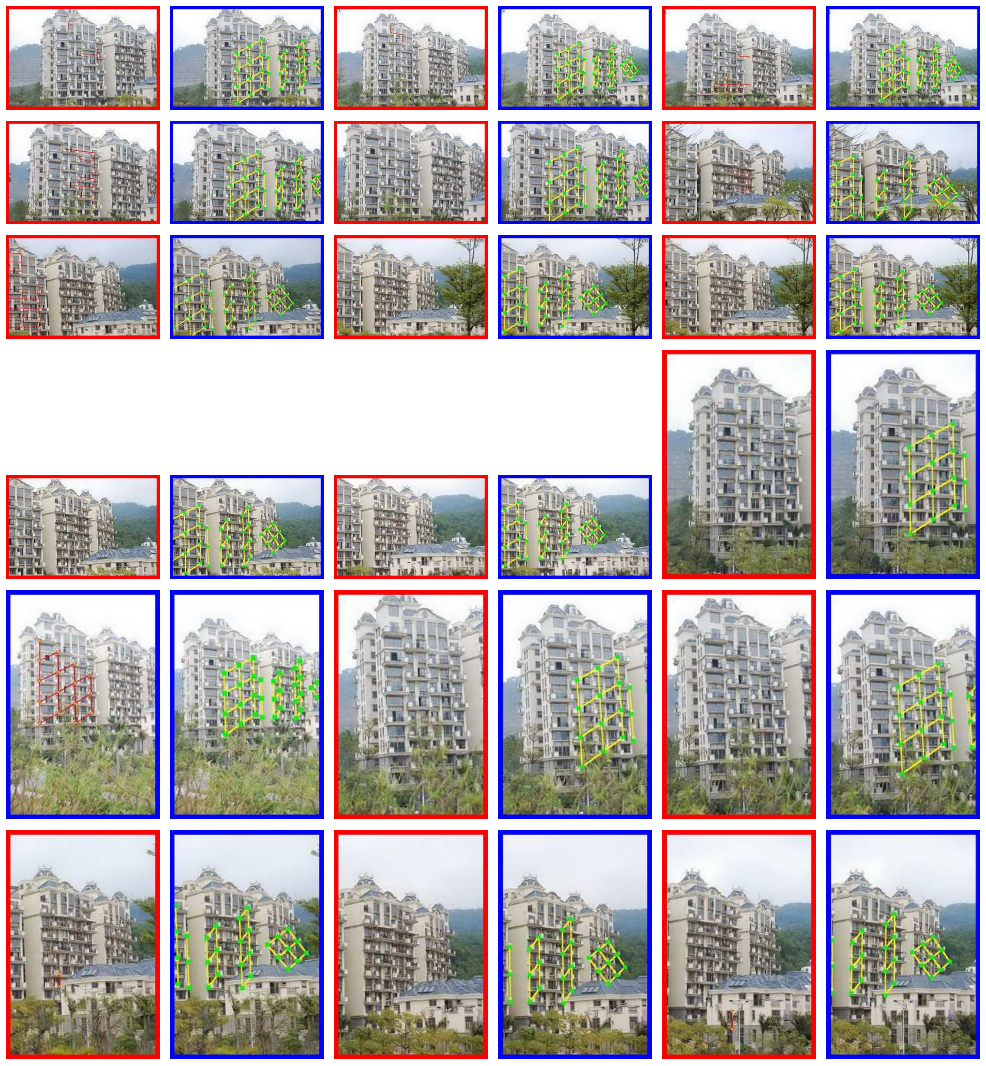


Table B.17: Lattice detection comparison with (67) on data 11.

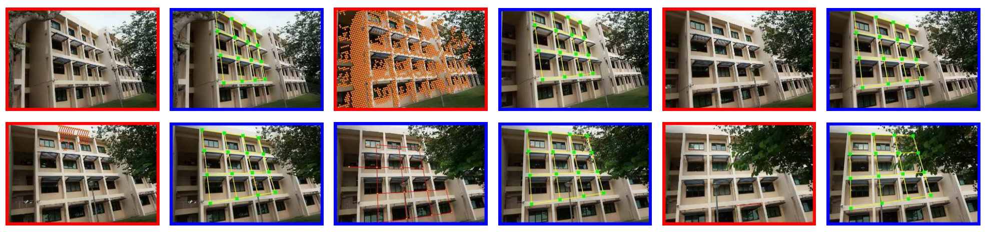


Table B.18: Lattice detection comparison with (67) on data 12.

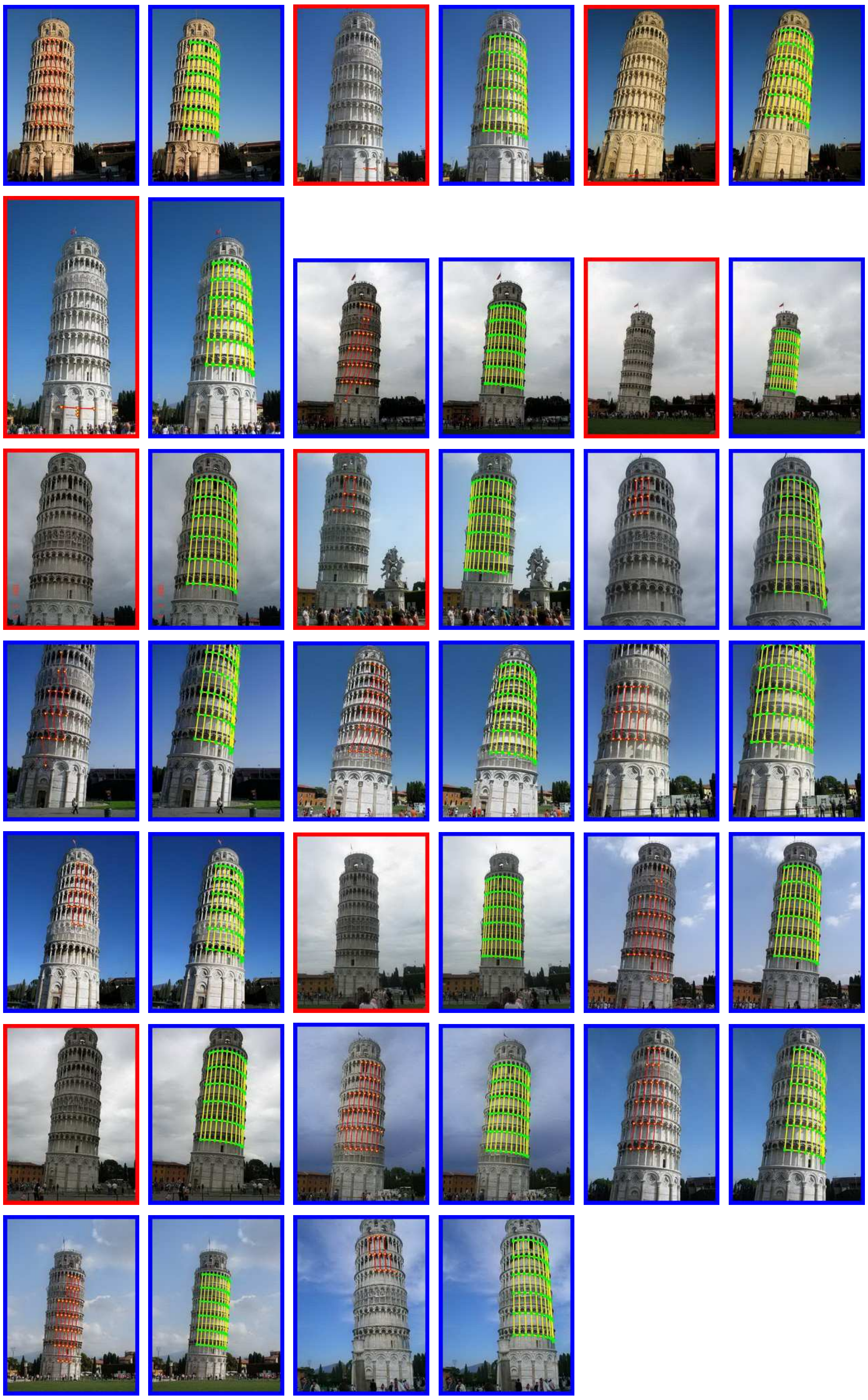


Table B.19: Lattice detection comparison with (67) on data 14.

Appendix C

Symmetry-based Stereo

The following figure shows the reconstructed 3D points from symmetry-based stereo. Each reconstructed 3D point is viewed as a patch with normal determined according to the frustum faces.

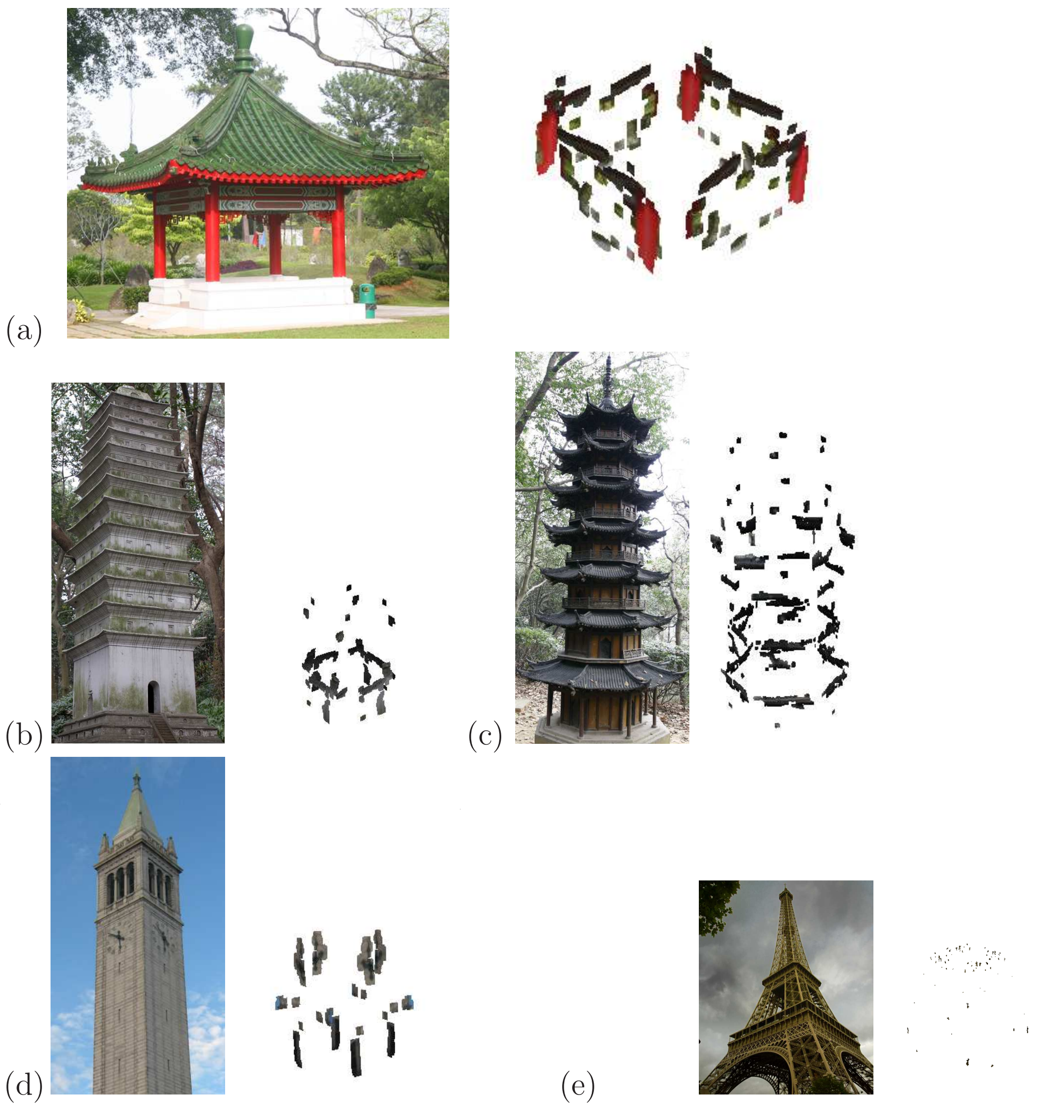


Figure C.1: Reconstructed 3D points from symmetry-based stereo. (a) The Pavilion example. (b) and (c) are two pagoda examples. (d) and (e) are results obtained for Berkeley Campanile and Eiffel Tower, reconstructed 3D points are relatively few for these two examples. This could be explained by matching failure on the textureless façade of Berkeley Campanile and bad correspondences between the rectified views of the curved façade of Eiffel Tower (which violates the homography transformation assumption between frustum faces).

Appendix D

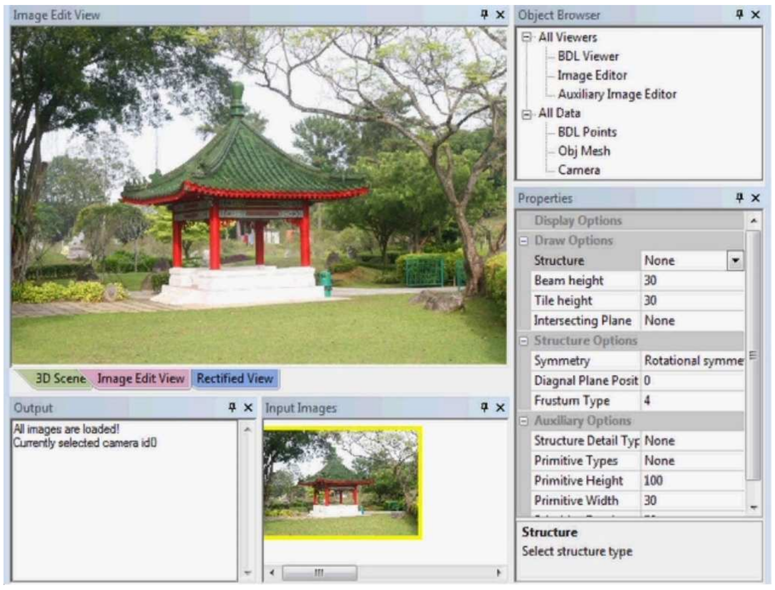
Modeling Interface

To assist efficient architecture modeling from a single image, we designed a prototype user interface that convert simple user clicks and strokes to 3D shapes. An overview of the interface is shown in Figure D.1

The user first need to click on the six frustum vertices to calibrate the camera. As shown in Figure D.2 (a), the frustum vertices are marked out as yellow and the calibrated camera parameters are shown in display area on the bottom left of the figure. With the camera parameters computed, virtual camera can be inserted at its symmetric position to enable stereo matching. Figure D.2 (b) shows the recovered 3D points on the pavilion.

The user starts modeling by marking out the wall planes as shown in Figure D.3.

To model the roof, the user marks out the roof silhouettes in the auxiliary planes in the image (Figure D.6 and Figure D.7), which are computed according to the frustum parameters recovered from camera calibration and the user strokes provided on the fly. The generated model is shown in Figure D.5.



All images are loaded!
Currently selected camera id0

Structure	None
Beam height	30
Tile height	30
Intersecting Plane	None
Symmetry	Rotational symme
Diagnal Plane Posit	0
Frustum Type	4
Structure Detail Typ	None
Primitive Types	None
Primitive Height	100
Primitive Width	30

Structure
Select structure type

Figure D.1: Overview of the user interface.

More details can be added to refine the model. For examples, the user can mark out the tiling pattern on the rectified frontal view of the roof (Figure D.8), insert pillars at symmetric positions (Figure D.9) and add revolved object to the roof top (Figure D.10).

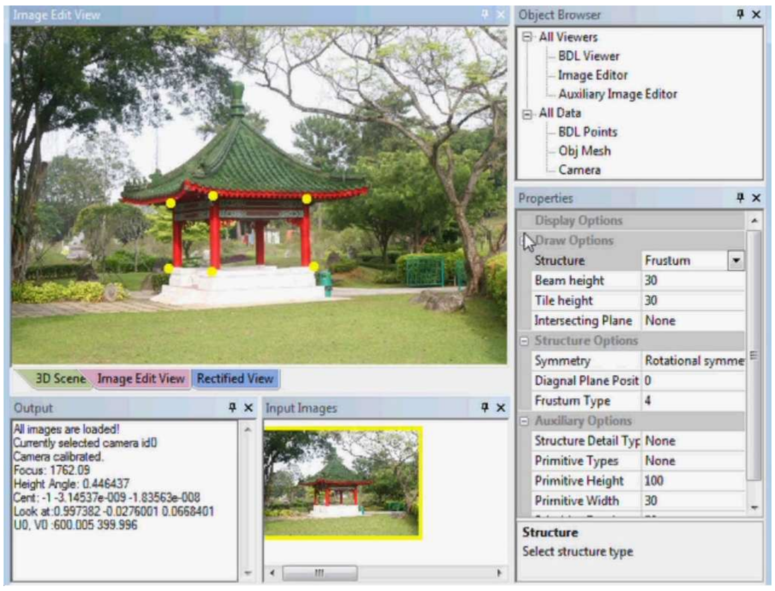
For buildings with multiple floors, the user can save the interaction by creating one reference floor as before and duplicate the rest by one stroke indicating the height for each floor on the image, see Figure D.12.

Similar auxiliary planes are also used for modeling architectures exhibiting bilateral symmetry, see Figure D.11. The orientation of these auxiliary planes can be adjusted by sliding a parameter bar to fit different architectures.

One addtional stroke is required for modeling roof of bilaterally symmetric architectures, since one cannot model all four sides of the roof by marking out just one side Figure D.13.

We summarize in Figure D.14 the user strokes used to create some of the

(a)

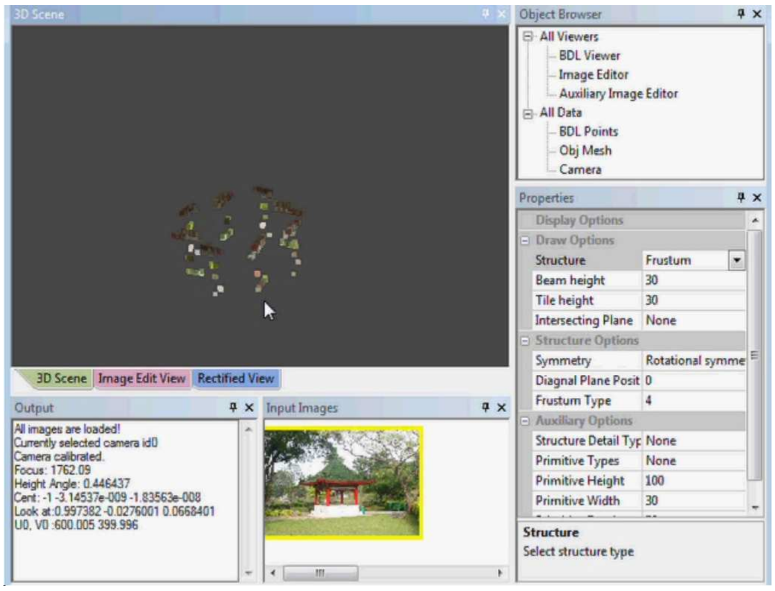


```
All images are loaded!
Currently selected camera id0
Camera calibrated.
Focus: 1762.09
Height Angle: 0.446437
Cent: -1 -3.14537e-009 -1.83563e-008
Look at:0.997382 -0.0276001 0.0668401
U0, V0 :600.005 399.996
```

Display Options	
Draw Options	
Structure	Frustum
Beam height	30
Tile height	30
Intersecting Plane	None
Structure Options	
Symmetry	Rotational symme
Diagnal Plane Posit	0
Frustum Type	4
Auxiliary Options	
Structure Detail Typ	None
Primitive Types	None
Primitive Height	100
Primitive Width	30

Structure
Select structure type

(b)

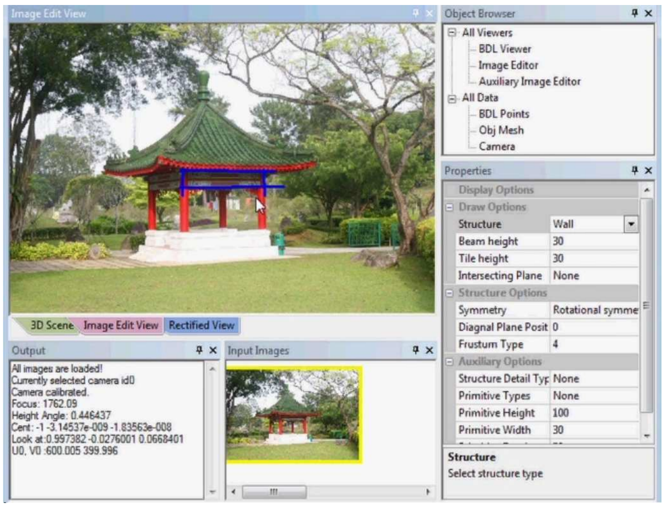


```
All images are loaded!
Currently selected camera id0
Camera calibrated.
Focus: 1762.09
Height Angle: 0.446437
Cent: -1 -3.14537e-009 -1.83563e-008
Look at:0.997382 -0.0276001 0.0668401
U0, V0 :600.005 399.996
```

Display Options	
Draw Options	
Structure	Frustum
Beam height	30
Tile height	30
Intersecting Plane	None
Structure Options	
Symmetry	Rotational symme
Diagnal Plane Posit	0
Frustum Type	4
Auxiliary Options	
Structure Detail Typ	None
Primitive Types	None
Primitive Height	100
Primitive Width	30

Structure
Select structure type

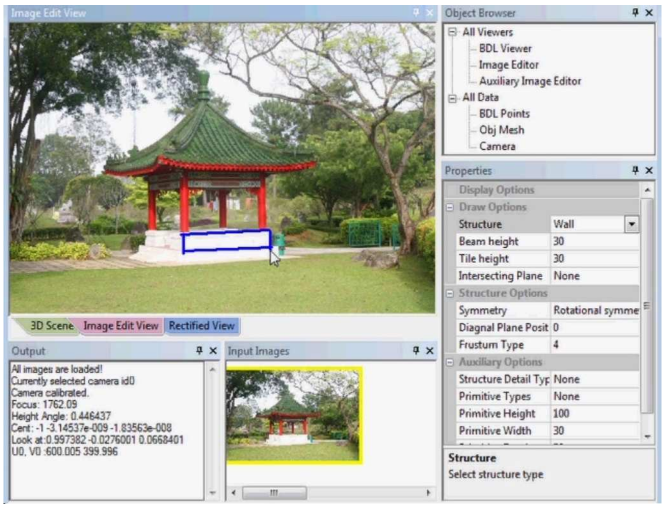
Figure D.2: (a) The user calibrates the camera with a single image interactively. (b) Reconstructed 3D points from stereo matching between the real view and the virtual view.

(a) 

```
All images are loaded!
Currently selected camera id0
Camera calibrated.
Focus: 1762.09
Height Angle: 0.446437
Cent: -1 -3.14537e-009 -1.83563e-008
Look at:0.997382 -0.0276001 0.0668401
U0, V0 :600.005 399.996
```

Display Options	
Draw Options	
Structure	Wall
Beam height	30
Tile height	30
Intersecting Plane	None
Structure Options	
Symmetry	Rotational symme
Diagnal Plane Posit	0
Frustum Type	4
Auxiliary Options	
Structure Detail Typ	None
Primitive Types	None
Primitive Height	100
Primitive Width	30

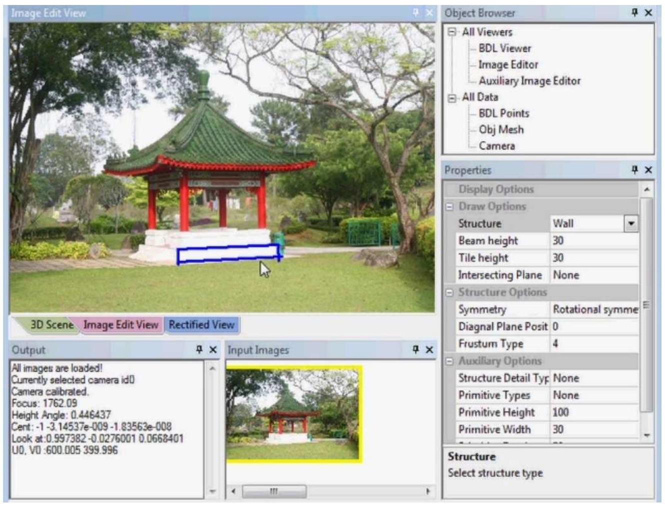
Structure
Select structure type

(b) 

```
All images are loaded!
Currently selected camera id0
Camera calibrated.
Focus: 1762.09
Height Angle: 0.446437
Cent: -1 -3.14537e-009 -1.83563e-008
Look at:0.997382 -0.0276001 0.0668401
U0, V0 :600.005 399.996
```

Display Options	
Draw Options	
Structure	Wall
Beam height	30
Tile height	30
Intersecting Plane	None
Structure Options	
Symmetry	Rotational symme
Diagnal Plane Posit	0
Frustum Type	4
Auxiliary Options	
Structure Detail Typ	None
Primitive Types	None
Primitive Height	100
Primitive Width	30

Structure
Select structure type

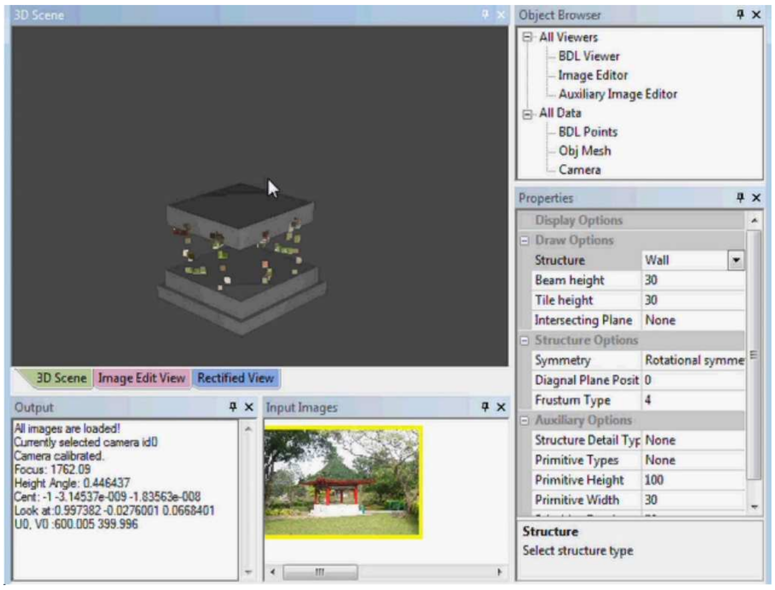
(c) 

```
All images are loaded!
Currently selected camera id0
Camera calibrated.
Focus: 1762.09
Height Angle: 0.446437
Cent: -1 -3.14537e-009 -1.83563e-008
Look at:0.997382 -0.0276001 0.0668401
U0, V0 :600.005 399.996
```

Display Options	
Draw Options	
Structure	Wall
Beam height	30
Tile height	30
Intersecting Plane	None
Structure Options	
Symmetry	Rotational symme
Diagnal Plane Posit	0
Frustum Type	4
Auxiliary Options	
Structure Detail Typ	None
Primitive Types	None
Primitive Height	100
Primitive Width	30

Structure
Select structure type

Figure D.3: The user models the wall planes by marking out the wall region. (a) and (b) show the user strokes used to create mesh object in Figure D.4.

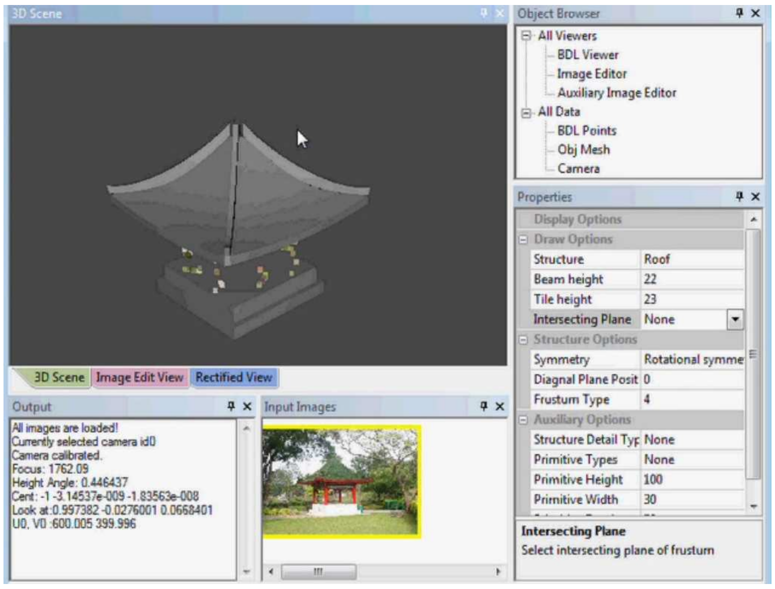


```
All images are loaded!
Currently selected camera id0
Camera calibrated.
Focus: 1762.09
Height Angle: 0.446437
Cent: -1 -3.14537e-009 -1.83563e-008
Look at:0.997382 -0.0276001 0.0668401
U0, V0 :600.005 399.996
```

Display Options	
Draw Options	
Structure	Wall
Beam height	30
Tile height	30
Intersecting Plane	None
Structure Options	
Symmetry	Rotational symme
Diagnal Plane Posit	0
Frustum Type	4
Auxiliary Options	
Structure Detail Typ	None
Primitive Types	None
Primitive Height	100
Primitive Width	30

Structure
Select structure type

Figure D.4: The wall model is created from the user strokes shown in Figure D.3.



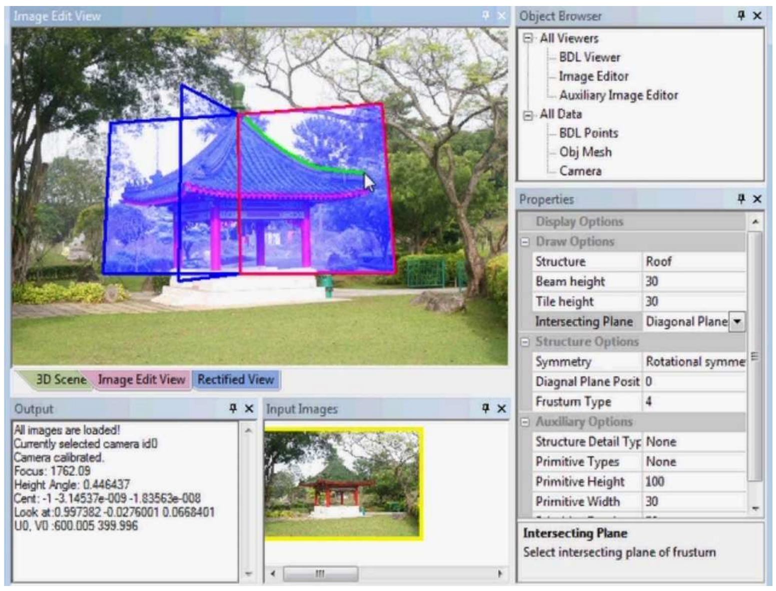
```
All images are loaded!
Currently selected camera id0
Camera calibrated.
Focus: 1762.09
Height Angle: 0.446437
Cent: -1 -3.14537e-009 -1.83563e-008
Look at:0.997382 -0.0276001 0.0668401
U0, V0 :600.005 399.996
```

Display Options	
Draw Options	
Structure	Roof
Beam height	22
Tile height	23
Intersecting Plane	None
Structure Options	
Symmetry	Rotational symme
Diagnal Plane Posit	0
Frustum Type	4
Auxiliary Options	
Structure Detail Typ	None
Primitive Types	None
Primitive Height	100
Primitive Width	30

Intersecting Plane
Select intersecting plane of frustum

Figure D.5: Roof model from the user strokes.

(a)

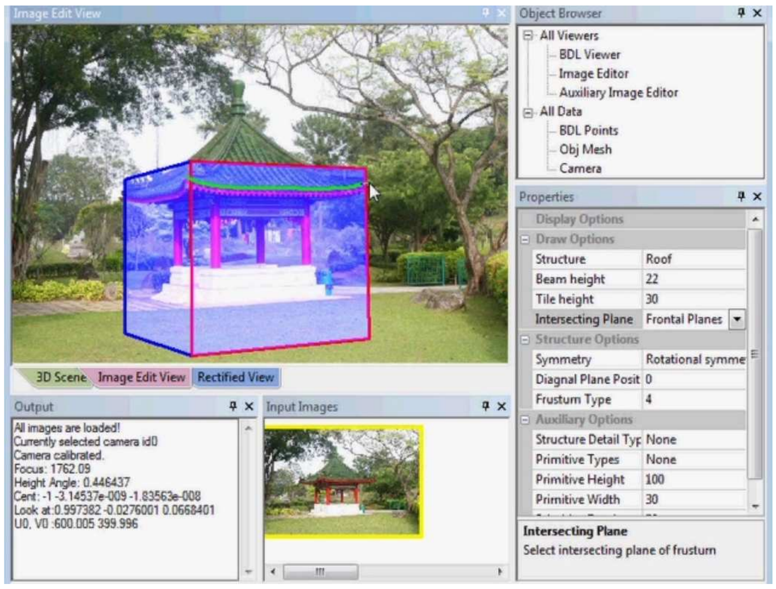


```
All images are loaded!
Currently selected camera id0
Camera calibrated.
Focus: 1762.09
Height Angle: 0.446437
Cent: -1 -3.14537e-009 -1.83563e-008
Look at:0.997382 -0.0276001 0.0668401
U0, V0 :600.005 399.996
```

Display Options	
Draw Options	
Structure	Roof
Beam height	30
Tile height	30
Intersecting Plane	Diagonal Plane
Structure Options	
Symmetry	Rotational symme
Diagnal Plane Posit	0
Frustum Type	4
Auxiliary Options	
Structure Detail Typ	None
Primitive Types	None
Primitive Height	100
Primitive Width	30

Intersecting Plane
Select intersecting plane of frustum

(b)



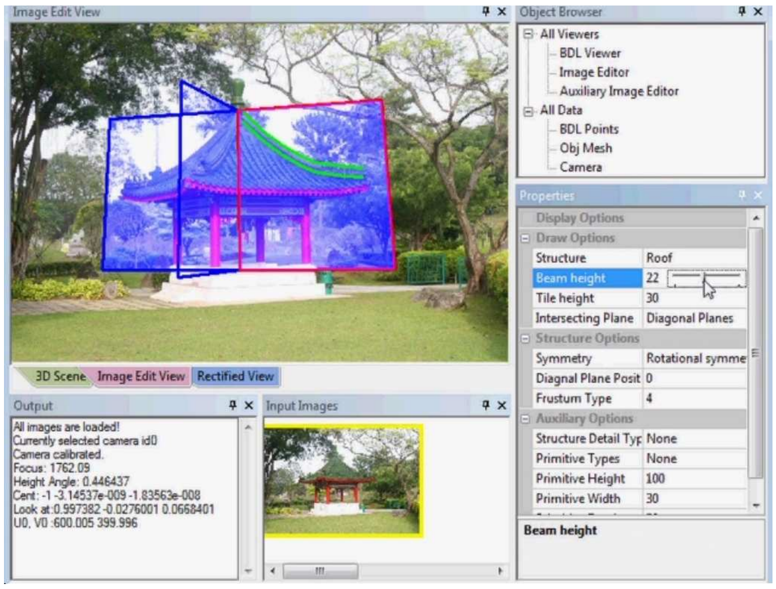
```
All images are loaded!
Currently selected camera id0
Camera calibrated.
Focus: 1762.09
Height Angle: 0.446437
Cent: -1 -3.14537e-009 -1.83563e-008
Look at:0.997382 -0.0276001 0.0668401
U0, V0 :600.005 399.996
```

Display Options	
Draw Options	
Structure	Roof
Beam height	22
Tile height	30
Intersecting Plane	Frontal Planes
Structure Options	
Symmetry	Rotational symme
Diagnal Plane Posit	0
Frustum Type	4
Auxiliary Options	
Structure Detail Typ	None
Primitive Types	None
Primitive Height	100
Primitive Width	30

Intersecting Plane
Select intersecting plane of frustum

Figure D.6: The user marks out the roof silhouettes in the auxiliary planes as shown in (a) and (b).

(a)

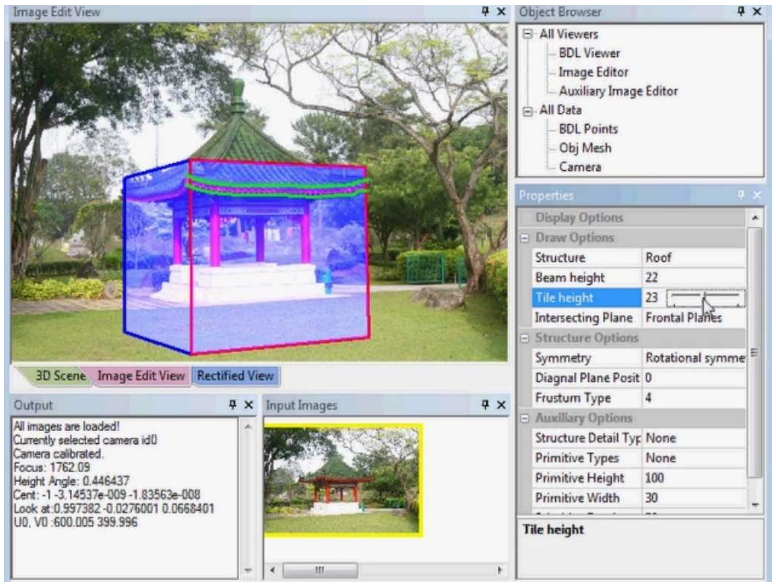


```
All images are loaded!
Currently selected camera id0
Camera calibrated.
Focus: 1762.09
Height Angle: 0.446437
Cent: -1 -3.14537e-009 -1.83563e-008
Look at:0.997382 -0.0276001 0.0668401
U0, V0 :600.005 399.996
```

Display Options	
Draw Options	
Structure	Roof
Beam height	22
Tile height	30
Intersecting Plane	Diagonal Planes
Structure Options	
Symmetry	Rotational symme
Diagnal Plane Posit	0
Frustum Type	4
Auxiliary Options	
Structure Detail Typ	None
Primitive Types	None
Primitive Height	100
Primitive Width	30

Beam height

(b)



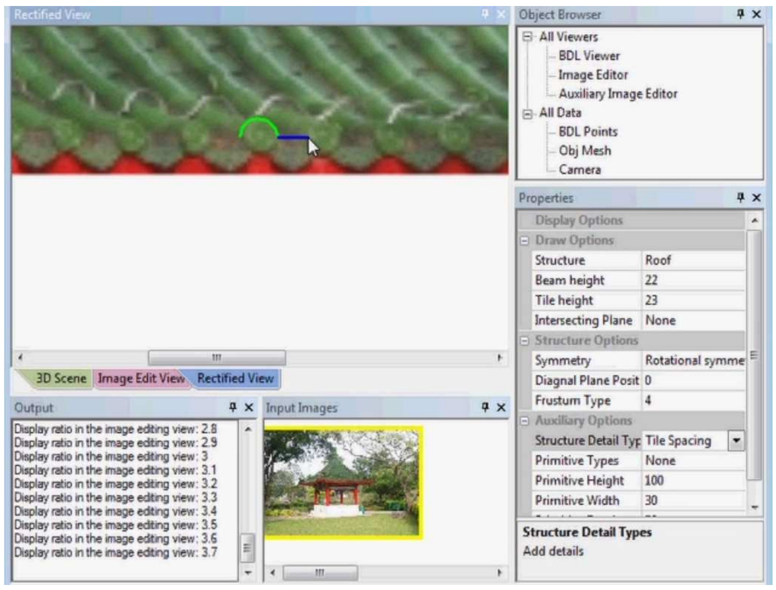
```
All images are loaded!
Currently selected camera id0
Camera calibrated.
Focus: 1762.09
Height Angle: 0.446437
Cent: -1 -3.14537e-009 -1.83563e-008
Look at:0.997382 -0.0276001 0.0668401
U0, V0 :600.005 399.996
```

Display Options	
Draw Options	
Structure	Roof
Beam height	22
Tile height	23
Intersecting Plane	Frontal Planes
Structure Options	
Symmetry	Rotational symme
Diagnal Plane Posit	0
Frustum Type	4
Auxiliary Options	
Structure Detail Typ	None
Primitive Types	None
Primitive Height	100
Primitive Width	30

Tile height

Figure D.7: The height of the roof beams can be adjusted by changing the corresponding parameters as shown in (c) and (d).

(a)

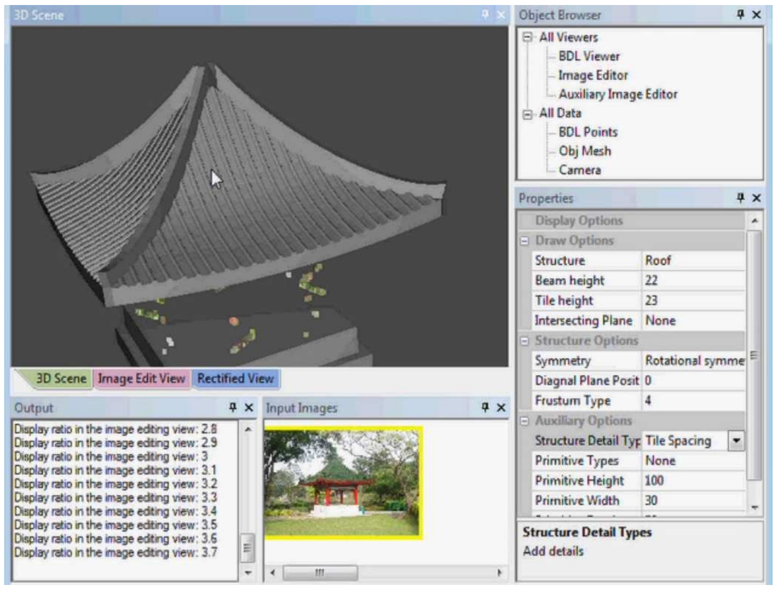


Display Options	
Draw Options	
Structure	Roof
Beam height	22
Tile height	23
Intersecting Plane	None
Structure Options	
Symmetry	Rotational symme
Diagnal Plane Posit	0
Frustum Type	4
Auxiliary Options	
Structure Detail Typ	Tile Spacing
Primitive Types	None
Primitive Height	100
Primitive Width	30

Structure Detail Types
Add details

```
Display ratio in the image editing view: 2.8
Display ratio in the image editing view: 2.9
Display ratio in the image editing view: 3
Display ratio in the image editing view: 3.1
Display ratio in the image editing view: 3.2
Display ratio in the image editing view: 3.3
Display ratio in the image editing view: 3.4
Display ratio in the image editing view: 3.5
Display ratio in the image editing view: 3.6
Display ratio in the image editing view: 3.7
```

(b)

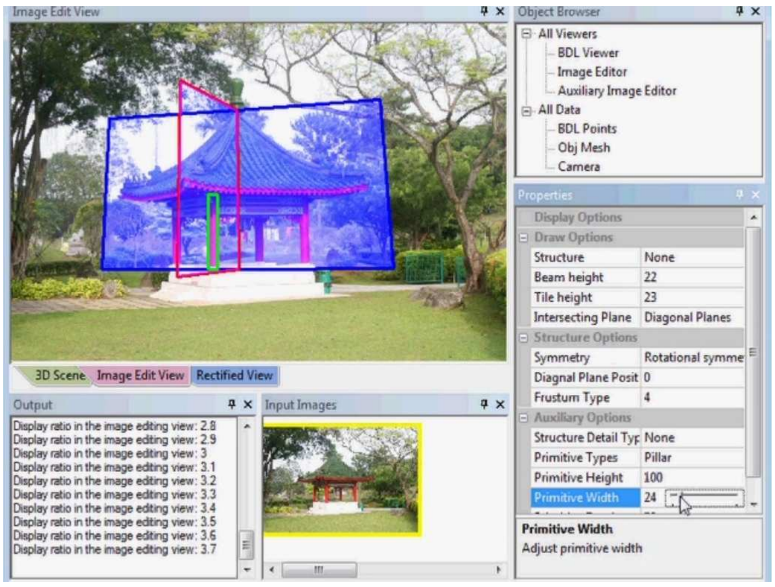


Display Options	
Draw Options	
Structure	Roof
Beam height	22
Tile height	23
Intersecting Plane	None
Structure Options	
Symmetry	Rotational symme
Diagnal Plane Posit	0
Frustum Type	4
Auxiliary Options	
Structure Detail Typ	Tile Spacing
Primitive Types	None
Primitive Height	100
Primitive Width	30

Structure Detail Types
Add details

```
Display ratio in the image editing view: 2.8
Display ratio in the image editing view: 2.9
Display ratio in the image editing view: 3
Display ratio in the image editing view: 3.1
Display ratio in the image editing view: 3.2
Display ratio in the image editing view: 3.3
Display ratio in the image editing view: 3.4
Display ratio in the image editing view: 3.5
Display ratio in the image editing view: 3.6
Display ratio in the image editing view: 3.7
```

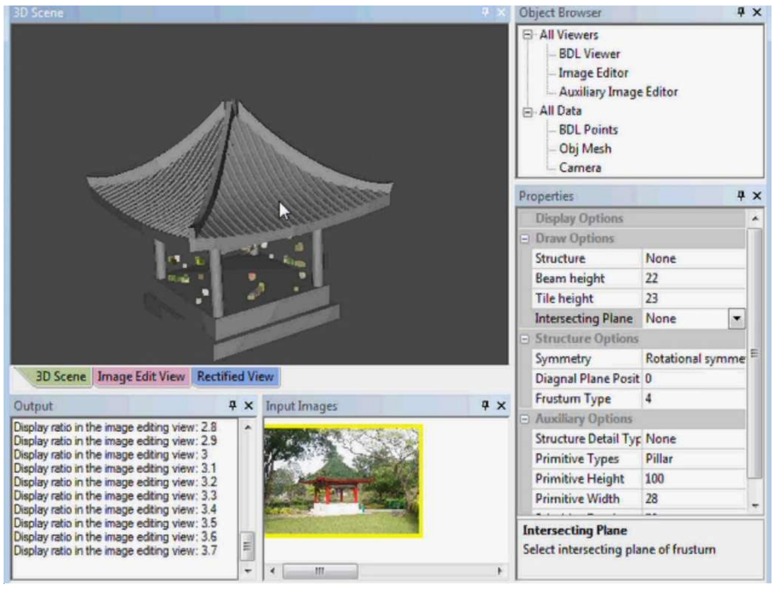
Figure D.8: (a) Apply tiling to the roof surface by marking out the tile pattern. (b) Refined roof model.

(a)


```
Display ratio in the image editing view: 2.8
Display ratio in the image editing view: 2.9
Display ratio in the image editing view: 3
Display ratio in the image editing view: 3.1
Display ratio in the image editing view: 3.2
Display ratio in the image editing view: 3.3
Display ratio in the image editing view: 3.4
Display ratio in the image editing view: 3.5
Display ratio in the image editing view: 3.6
Display ratio in the image editing view: 3.7
```

Structure	None
Beam height	22
Tile height	23
Intersecting Plane	Diagonal Planes
Symmetry	Rotational symme
Diagnal Plane Posit	0
Frustum Type	4
Structure Detail Typ	None
Primitive Types	Pillar
Primitive Height	100
Primitive Width	24

Primitive Width
Adjust primitive width

(b)


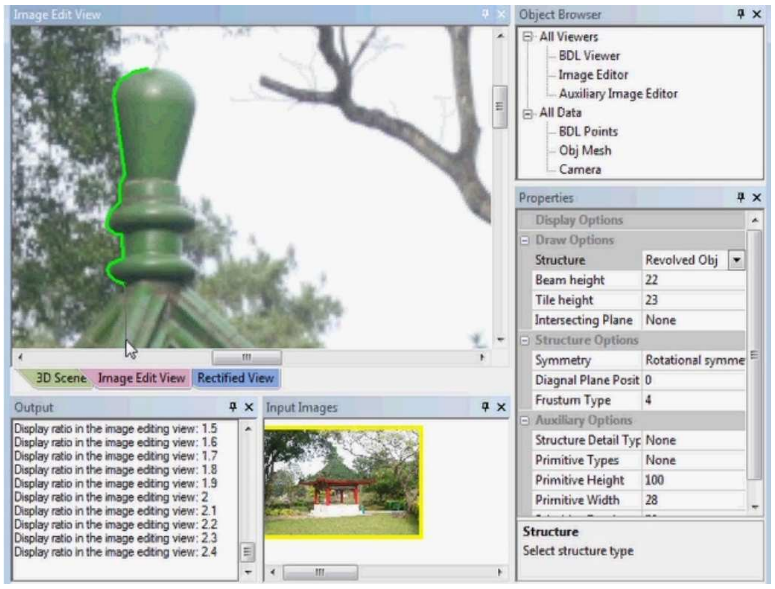
```
Display ratio in the image editing view: 2.8
Display ratio in the image editing view: 2.9
Display ratio in the image editing view: 3
Display ratio in the image editing view: 3.1
Display ratio in the image editing view: 3.2
Display ratio in the image editing view: 3.3
Display ratio in the image editing view: 3.4
Display ratio in the image editing view: 3.5
Display ratio in the image editing view: 3.6
Display ratio in the image editing view: 3.7
```

Structure	None
Beam height	22
Tile height	23
Intersecting Plane	None
Symmetry	Rotational symme
Diagnal Plane Posit	0
Frustum Type	4
Structure Detail Typ	None
Primitive Types	Pillar
Primitive Height	100
Primitive Width	28

Intersecting Plane
Select intersecting plane of frustum

Figure D.9: (a) Insert pillars at symmetric positions by specifying the height and diameter of the pillars. (b) Pavilion model with inserted pillars.

(a)

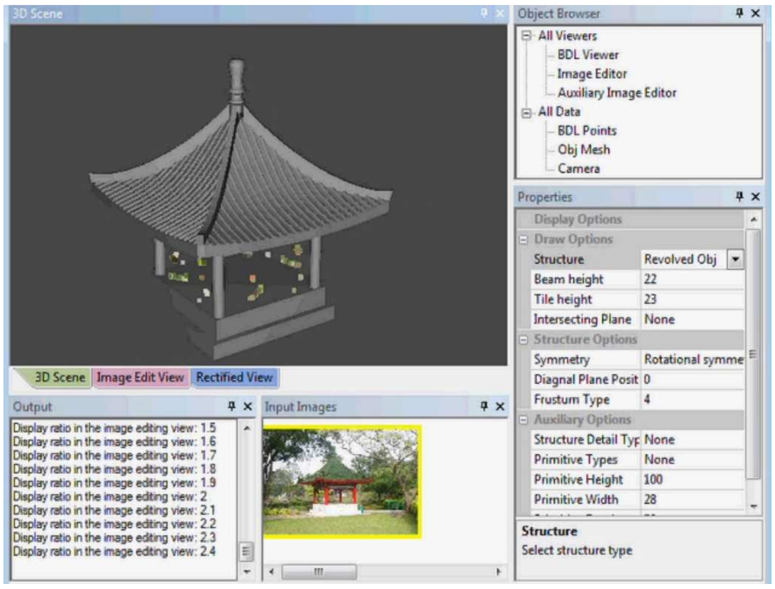


```
Display ratio in the image editing view: 1.5
Display ratio in the image editing view: 1.6
Display ratio in the image editing view: 1.7
Display ratio in the image editing view: 1.8
Display ratio in the image editing view: 1.9
Display ratio in the image editing view: 2
Display ratio in the image editing view: 2.1
Display ratio in the image editing view: 2.2
Display ratio in the image editing view: 2.3
Display ratio in the image editing view: 2.4
```

Display Options	
Draw Options	
Structure	Revolved Obj
Beam height	22
Tile height	23
Intersecting Plane	None
Structure Options	
Symmetry	Rotational symme
Diagnal Plane Posit	0
Frustum Type	4
Auxiliary Options	
Structure Detail Typ	None
Primitive Types	None
Primitive Height	100
Primitive Width	28

Structure
Select structure type

(b)



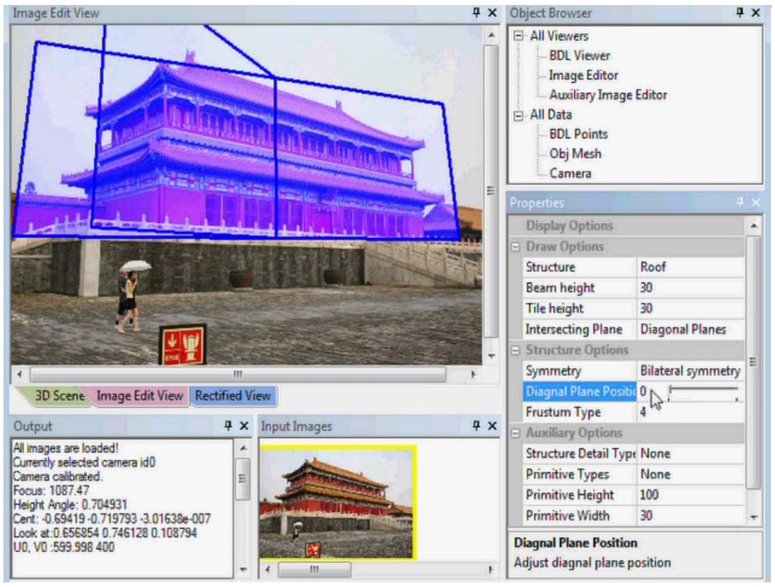
```
Display ratio in the image editing view: 1.5
Display ratio in the image editing view: 1.6
Display ratio in the image editing view: 1.7
Display ratio in the image editing view: 1.8
Display ratio in the image editing view: 1.9
Display ratio in the image editing view: 2
Display ratio in the image editing view: 2.1
Display ratio in the image editing view: 2.2
Display ratio in the image editing view: 2.3
Display ratio in the image editing view: 2.4
```

Display Options	
Draw Options	
Structure	Revolved Obj
Beam height	22
Tile height	23
Intersecting Plane	None
Structure Options	
Symmetry	Rotational symme
Diagnal Plane Posit	0
Frustum Type	4
Auxiliary Options	
Structure Detail Typ	None
Primitive Types	None
Primitive Height	100
Primitive Width	28

Structure
Select structure type

Figure D.10: (a) Insert revolved object by marking the object silhouette. (b) Pavilion model with revolved object.

(a)



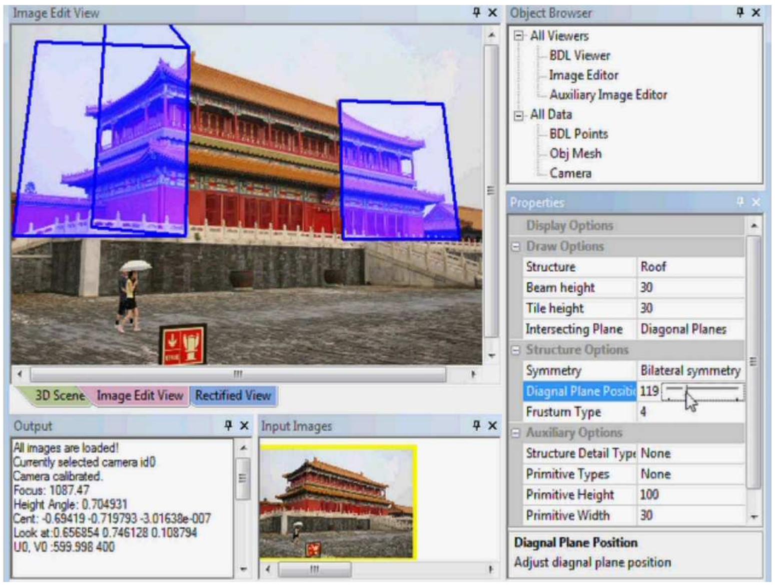
Output

```
All images are loaded!
Currently selected camera id0
Camera calibrated.
Focus: 1087.47
Height Angle: 0.704931
Cent: -0.69419 -0.719793 -3.01638e-007
Look at:0.656854 0.746128 0.108794
U0, V0 :599.998 400
```

Display Options	
Draw Options	
Structure	Roof
Beam height	30
Tile height	30
Intersecting Plane	Diagonal Planes
Structure Options	
Symmetry	Bilateral symmetry
Diagnal Plane Positi	0
Frustum Type	4
Auxiliary Options	
Structure Detail Type	None
Primitive Types	None
Primitive Height	100
Primitive Width	30

Diagnal Plane Position
Adjust diagnal plane position

(b)



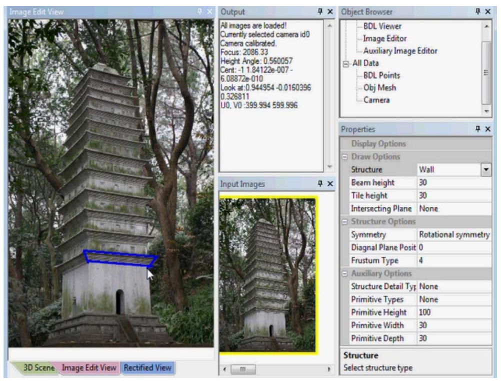
Output

```
All images are loaded!
Currently selected camera id0
Camera calibrated.
Focus: 1087.47
Height Angle: 0.704931
Cent: -0.69419 -0.719793 -3.01638e-007
Look at:0.656854 0.746128 0.108794
U0, V0 :599.998 400
```

Display Options	
Draw Options	
Structure	Roof
Beam height	30
Tile height	30
Intersecting Plane	Diagonal Planes
Structure Options	
Symmetry	Bilateral symmetry
Diagnal Plane Positi	119
Frustum Type	4
Auxiliary Options	
Structure Detail Type	None
Primitive Types	None
Primitive Height	100
Primitive Width	30

Diagnal Plane Position
Adjust diagnal plane position

Figure D.11: (a) Auxiliary planes computed from frustum parameters. (b) User adjusted auxiliary planes for this particular building.



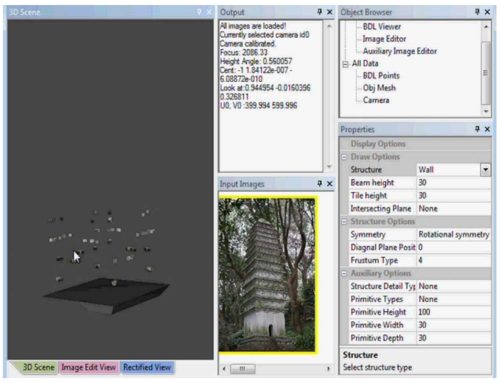
```
All images are loaded!
Currently selected camera id0
Camera calibrated.
Focus: 2086.33
Height Angle: 0.560057
Cent: -1 1.84122e-007 -
6.08872e-010
Look at:0.944954 -0.0160396
0.326811
U0, V0 :399.994 599.996
```

- BDL Viewer
- Image Editor
- Auxiliary Image Editor
- All Data
  - BDL Points
  - Obj Mesh
  - Camera

Display Options	
Draw Options	
Structure	Wall
Beam height	30
Tile height	30
Intersecting Plane	None
Structure Options	
Symmetry	Rotational symmetry
Diagnal Plane Posit	0
Frustum Type	4
Auxiliary Options	
Structure Detail Typ	None
Primitive Types	None
Primitive Height	100
Primitive Width	30
Primitive Depth	30

Structure
Select structure type

(a)



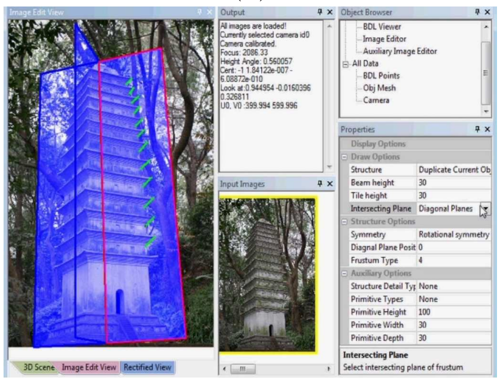
```
All images are loaded!
Currently selected camera id0
Camera calibrated.
Focus: 2086.33
Height Angle: 0.560057
Cent: -1 1.84122e-007 -
6.08872e-010
Look at:0.944954 -0.0160396
0.326811
U0, V0 :399.994 599.996
```

- BDL Viewer
- Image Editor
- Auxiliary Image Editor
- All Data
  - BDL Points
  - Obj Mesh
  - Camera

Display Options	
Draw Options	
Structure	Wall
Beam height	30
Tile height	30
Intersecting Plane	None
Structure Options	
Symmetry	Rotational symmetry
Diagnal Plane Posit	0
Frustum Type	4
Auxiliary Options	
Structure Detail Typ	None
Primitive Types	None
Primitive Height	100
Primitive Width	30
Primitive Depth	30

Structure
Select structure type

(b)



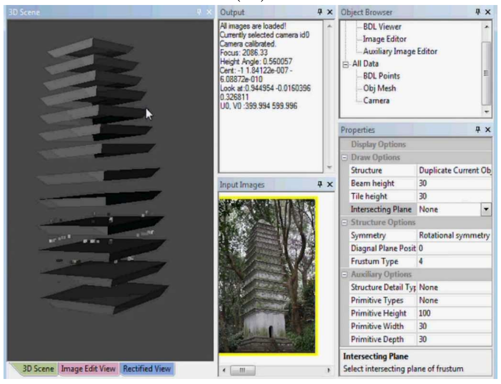
```
All images are loaded!
Currently selected camera id0
Camera calibrated.
Focus: 2086.33
Height Angle: 0.560057
Cent: -1 1.84122e-007 -
6.08872e-010
Look at:0.944954 -0.0160396
0.326811
U0, V0 :399.994 599.996
```

- BDL Viewer
- Image Editor
- Auxiliary Image Editor
- All Data
  - BDL Points
  - Obj Mesh
  - Camera

Display Options	
Draw Options	
Structure	Duplicate Current Ob
Beam height	30
Tile height	30
Intersecting Plane	Diagonal Planes
Structure Options	
Symmetry	Rotational symmetry
Diagnal Plane Posit	0
Frustum Type	4
Auxiliary Options	
Structure Detail Typ	None
Primitive Types	None
Primitive Height	100
Primitive Width	30
Primitive Depth	30

Intersecting Plane
Select intersecting plane of frustum

(c)



```
All images are loaded!
Currently selected camera id0
Camera calibrated.
Focus: 2086.33
Height Angle: 0.560057
Cent: -1 1.84122e-007 -
6.08872e-010
Look at:0.944954 -0.0160396
0.326811
U0, V0 :399.994 599.996
```

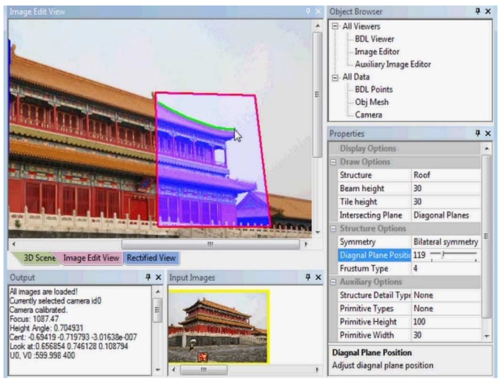
- BDL Viewer
- Image Editor
- Auxiliary Image Editor
- All Data
  - BDL Points
  - Obj Mesh
  - Camera

Display Options	
Draw Options	
Structure	Duplicate Current Ob
Beam height	30
Tile height	30
Intersecting Plane	None
Structure Options	
Symmetry	Rotational symmetry
Diagnal Plane Posit	0
Frustum Type	4
Auxiliary Options	
Structure Detail Typ	None
Primitive Types	None
Primitive Height	100
Primitive Width	30
Primitive Depth	30

Intersecting Plane
Select intersecting plane of frustum

(d)

Figure D.12: (a) User strokes for creating the reference floor. (b) Floor model and reconstructed 3D points from stereo matching. (c) User strokes for floor duplication. (d) Multiple floor models obtained by translating and resizing the reference floor according to the user strokes in (c).

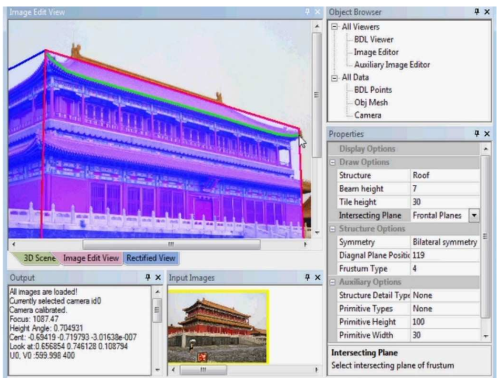


Display Options	
Draw Options	
Structure	Roof
Beam height	30
Tile height	30
Intersecting Plane	Diagonal Planes
Structure Options	
Symmetry	Bilateral symmetry
Diagnal Plane Positic	119
Frustum Type	4
Auxiliary Options	
Structure Detail Type	None
Primitive Types	None
Primitive Height	100
Primitive Width	30

Diagnal Plane Position
Adjust diagnal plane position

```
All images are loaded!
Currently selected camera id0
Camera calibrated.
Focus: 1087.47
Height Angle: 0.704931
Cent: -0.69419 -0.719793 -3.01638e-007
Look at:0.656854 0.746128 0.108794
U0, V0 :599.998 400
```

(a)

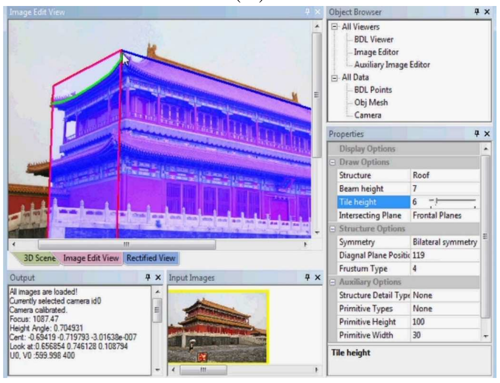


Display Options	
Draw Options	
Structure	Roof
Beam height	7
Tile height	30
Intersecting Plane	Frontal Planes
Structure Options	
Symmetry	Bilateral symmetry
Diagnal Plane Positic	119
Frustum Type	4
Auxiliary Options	
Structure Detail Type	None
Primitive Types	None
Primitive Height	100
Primitive Width	30

Intersecting Plane
Select intersecting plane of frustum

```
All images are loaded!
Currently selected camera id0
Camera calibrated.
Focus: 1087.47
Height Angle: 0.704931
Cent: -0.69419 -0.719793 -3.01638e-007
Look at:0.656854 0.746128 0.108794
U0, V0 :599.998 400
```

(b)

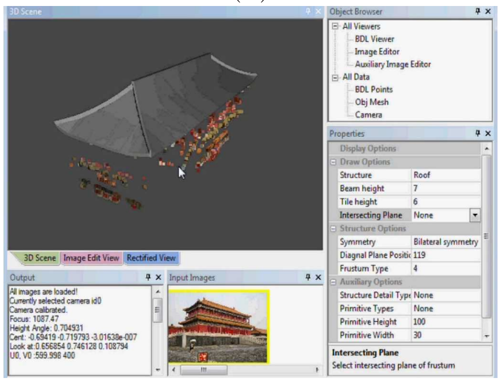


Display Options	
Draw Options	
Structure	Roof
Beam height	7
Tile height	6
Intersecting Plane	Frontal Planes
Structure Options	
Symmetry	Bilateral symmetry
Diagnal Plane Positic	119
Frustum Type	4
Auxiliary Options	
Structure Detail Type	None
Primitive Types	None
Primitive Height	100
Primitive Width	30

Tile height

```
All images are loaded!
Currently selected camera id0
Camera calibrated.
Focus: 1087.47
Height Angle: 0.704931
Cent: -0.69419 -0.719793 -3.01638e-007
Look at:0.656854 0.746128 0.108794
U0, V0 :599.998 400
```

(c)



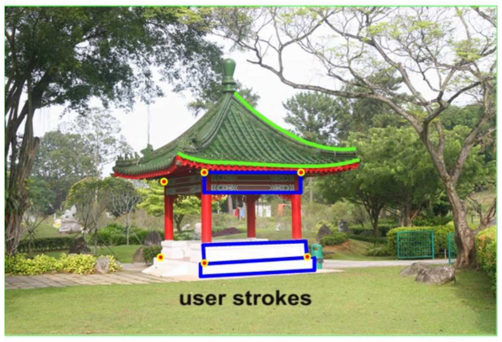
Display Options	
Draw Options	
Structure	Roof
Beam height	7
Tile height	6
Intersecting Plane	None
Structure Options	
Symmetry	Bilateral symmetry
Diagnal Plane Positic	119
Frustum Type	4
Auxiliary Options	
Structure Detail Type	None
Primitive Types	None
Primitive Height	100
Primitive Width	30

Intersecting Plane
Select intersecting plane of frustum

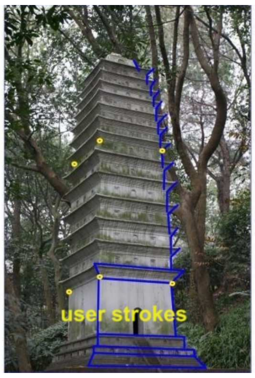
```
All images are loaded!
Currently selected camera id0
Camera calibrated.
Focus: 1087.47
Height Angle: 0.704931
Cent: -0.69419 -0.719793 -3.01638e-007
Look at:0.656854 0.746128 0.108794
U0, V0 :599.998 400
```

(d)

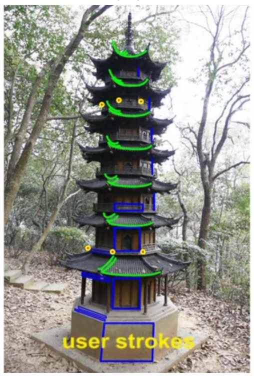
Figure D.13: (a), (b) and (c) are user strokes for creating the roof model in (d).



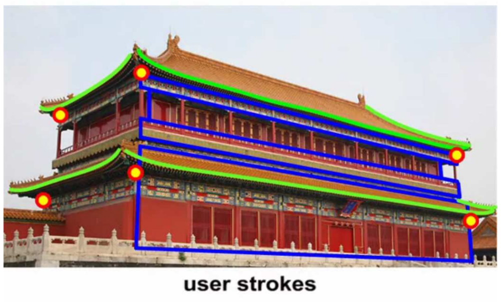
(a)



(b)



(c)



(d)

Figure D.14: (a) User strokes for creating pavilion model in Figure 5.1. (b) User strokes for creating pagoda model in Figure 5.11. (c) User strokes for creating pagoda model in Figure 5.12. (d) User strokes for creating pavilion model in Figure 5.2.

representitive 3D architecture models reported in Chapter 5.

Bibliography

[1] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski. Building rome in a day. In *Proc. ICCV*, 2009. 25, 111

[2] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256, 1992. 68

[3] C. Bibby and I. Reid. Simultaneous localisation and mapping in dynamic environments (SLAMIDE) with reversible data association. In *Proc. of Robotics Sci. and Syst.*, 2007. 32

[4] M. Bokeloh, A. Berner, M. Wand, H.-P. Seidel, and A. Schilling. Symmetry detection using feature lines. *Computer Graphics Forum*, 28, 2009. 5, 57

[5] M. Bokeloh, M. Wand, and H.-P. Seidel. A connection between partial symmetry and inverse procedural modeling. *ACM Trans. on Graph. (Proc. of SIGGRAPH)*, 29, 2010. 53, 57

[6] C. Bregler, A. Hertzmann, and H. Biermann. Recovering non-rigid 3d shape from image streams. In *Proc. CVPR*, volume 2, pages 690–696, 2000. 3

[7] R. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–159, 1991. 1

[8] A. M. Buchanan and A. W. Fitzgibbon. Damped newton algorithms for matrix factorization with missing data. In *Proc. CVPR*, pages 316–322,

2005. 22

[9] A. L. Chauve, P. Labatut, and J. P. Pons. Robust piecewise-planar 3d reconstruction and completion from large-scale unstructured point data. In *Proc. CVPR*, 2010. 5, 9, 71

[10] A. Cohen, C. Zach, S. Sinha, and M. Pollefeys. Discovering and exploiting 3d symmetries in structure from motion. In *Proc. CVPR*, 2012. 53, 75, 112

[11] B. Combes, R. Hennessy, J. Waddington, N. Roberts, and S. Prima. Automatic symmetry plane estimation of bilateral objects in point clouds. In *Proc. CVPR*, 2008. 5, 55, 57

[12] H. Cornelius and G. Loy. Detecting bilateral symmetry in perspective. In *Proc. of CVPR Workshop*, 2006. 5, 56

[13] D. Crandall, A. Owens, N. Snavely, and D. Huttenlocher. Discrete-continuous optimization for large-scale structure from motion. In *Proc. CVPR*, 2011. 25, 31, 111

[14] A. J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proc. ICCV (2)*, pages 1403–1410, 2003. 3

[15] P. Debevec. *Modeling and Rendering Architecture from Photographs.* University of California at Berkeley, Computer Science Division, Berkeyly CA, 1996. 82, 83, 85, 101

[16] P. Debevec, C. Taylor, and J. Malik. Modeling and rendering architecture from photographs: a hybrid geometry and image-based approach. In *Proc. ACM SIGGRAPH*, 1996. 77, 82, 84, 104

[17] A. R. Dick, P. H. S. Torr, and R. Cipolla. Modelling and interpretation of architecture from several images. *Int. J. Comput. Vision*, 60:111–134, 2004. 84

[18] O. Enqvist, F. Kahl, and C. Olsson. Non-sequential structure from motion. In *Proc. ICCV Workshops*, pages 264–271, 2011. 28

[19] Fischler, A. Martin, and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartograph. *Commun. ACM*, 24(6):381–395, 1981. 37, 60

[20] A. W. Fitzgibbon, G. Cross, and A. Zisserman. Automatric 3d model construction for turn-table sequences. In *In Proc. of SMILE Workshop on Structure from Multiple Images in Large Scale Environments*, pages 155–170, 1998. 90

[21] A. Francois, G. Medioni, and R. Waupotitsch. Reconstructing mirror symmetric scenes from a single view using 2-view stereo geometry. In *In Proc. of ICPR*, 2002. 9, 79, 81, 92

[22] W. Freeman, E. Pasztor, and O. Carmichael. Learning low-level vision. *Int. J. Comput. Vision*, 2000. 98

[23] C. Früh and A. Zakhor. Constructing 3d city models by merging ground-based and airborne views. In *Proc. CVPR*, 2003. 84

[24] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32:1362–1376, 2010. 2, 32, 38

[25] A. Gil, O. Reinoso, O. Mozos, C. Stachnissi, and W. Burgard. Improving data association in vision-based slam. In *Proc. IROS*, pages 2076–2081, 2006. 32

[26] V. Govindu. Combining two-view constraints for motion estimation. In *Proc. CVPR*, pages 218–225, 2001. 23

[27] V. Govindu. Robustness in motion averaging. In *Proc. ACCV*, pages 457–

466, 2006. 31

[28] I. Hargittai and M. Hargittai. *Symmetry: A Unifying Concept*. Shelter Publications, 1994. 79

[29] C. Harris and M. Stephens. A combined corner and edge detector. In *Proc. Alvey Vision Conference*, pages 147–151, 1988. 3

[30] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004. 3, 16, 17, 19, 21, 85, 88, 92

[31] M. Havlena, A. Torii, J. Knopp, and T. Pajdla. Randomized structure from motion based on atomic 3d models from camera triplets. In *Proc. CVPR*, pages 2874–2881, 2009. 28, 31

[32] J. Hays, M. Leordeanu, A. A. Efros, and Y. Liu. Discovering texture regularity as a higher-order correspondence problem. In *Proc. ECCV*, 2006. 5, 54, 56, 58

[33] W. Hong, A. Yang, K. Huang, and Y. Ma. On symmetry and multiple-view geometry: Structure, pose, and calibration from a single image. *Int. J. Comput. Vision*, pages 241–265, 2004. 9, 79, 81, 82, 92

[34] B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981. 3

[35] N. Jiang, P. Tan, and L.-F. Cheong. Symmetric architecture modeling with a single image. *ACM Trans. on Graph. (Proc. of SIGGRAPH Asia)*, 28(5), 2009. 7

[36] N. Jiang, P. Tan, and L.-F. Cheong. Multi-view repetitive structure detection. In *Proc. ICCV*, pages 535–542, 2011. 7

[37] N. Jiang, P. Tan, and L.-F. Cheong. Seeing double without confusion:

Structure-from-motion in highly ambiguous scenes. In *Proc. CVPR*, pages 1458–1465, 2012. 7

[38] F. Kahl. Multiple view geometry and the l-infinity norm. In *Proc. ICCV*, 2005. 24

[39] Q. Ke and T. Kanade. Robust l" norm factorization in the presence of outliers and missing data by alternative convex programming. In *In Proc. CVPR - Volume 1*, pages 739–746, 2005. 22

[40] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *Proc. International Symposium on Mixed and Augmented Reality*, pages 1–10, 2007. 2, 3

[41] M. Klopschitz, A. Irschara, G. Reitmayr, and D. Schmalstieg. Robust incremental structure from motion. In *Proc. 3DPVT*, 2010. 25, 31

[42] T. Korah and C. Rasmussen. Analysis of building textures for reconstructing partially occluded façades. In *Proc. ECCV*, 2008. 58

[43] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: Image and video synthesis using graph cuts. *ACM Trans. on Graph. (Proc. of SIGGRAPH)*, pages 277–286, 2003. 98

[44] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(2):150–162, 1994. 4

[45] S. Lee, R. T. Collins, and Y. Liu. Rotation symmetry group detection via frequency analysis of frieze-expansions. In *Proc. CVPR*, 2008. 5, 54, 56

[46] S. Lee and Y. Liu. Skewed rotation symmetry group detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2010. 5, 54

[47] M. Lhuillier and L. Quan. Match propagation for image-based modeling

and rendering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(8):1140–1146, 2002. 58, 92

[48] X. Li, C. Wu, C. Zach, S. Lazebnik, and J.-M. Frahm. Modeling and recognition of landmark image collections using iconic scene graphs. In *Proc. ECCV*, 2008. 25, 31

[49] D. Liebowitz, A. Criminisi, and A. Zisserman. Creating architectural models from images. *Computer Graphics Forum*, pages 39–50, 9 1999. 77, 83

[50] Y. Liu, R. T. Collins, and Y. Tsin. A computational model for periodic pattern perception based on frieze and wallpaper groups. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26:354–371, 2004. 5, 54, 56

[51] Y. Liu, J. H. Hays, Y. Xu, and H. Shum. Digital papercutting. In *SIGGRAPH Technical Sketch*, 2005. 5, 56

[52] Y. Liu, W.-C. Lin, and J. Hays. Near-regular texture analysis and manipulation. *ACM Trans. on Graph.*, 23:368–376, August 2004. 5, 54

[53] D. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60:91–110, 2004. 2, 3, 32, 59, 92

[54] G. Loy and J. Eklundh. Detecting symmetry and symmetric constellations of features. In *Proc. ECCV*, 2006. 5, 56

[55] D. Marr and T. Poggio. A Computational Theory of Human Stereo Vision. *Proc. Royal Society of London. Series B, Biological Sciences*, 204(1156):301–328, 1979. 2

[56] D. Martinec and T. Pajdla. Robust rotation and translation estimation in multiview reconstruction. In *Proc. CVPR*, 2007. 24, 28, 31, 42

[57] N. J. Mitra, L. J. Guibas, and M. Pauly. Partial and approximate symmetry detection for 3d geometry. *ACM Trans. on Graph. (Proc. of SIGGRAPH)*,

2006. 5, 55, 57

[58] P. Müller, P. Wonka, S. Haegler, A. Ulmer, and L. Van Gool. Procedural modeling of buildings. *Proc. ACM SIGGRAPH*, 25(3):614–623, 2006. 5, 77, 82

[59] P. Müller, G. Zeng, P. Wonka, and L. Van Gool. Image-based procedural modeling of façades. *ACM Trans. on Graph. (Proc. of SIGGRAPH)*, 26(85), 2007. 58, 83, 93

[60] L. Nan, A. Sharf, H. Zhang, D. Cohen-Or, and B. Chen. Smartboxes for interactive urban reconstruction. *ACM Trans. on Graph. (Proc. of SIGGRAPH)*, 2010. 5, 53, 58, 59, 68

[61] R. Newcombe, S. Lovegrove, and A. Davison. DTAM: Dense Tracking and Mapping in Real-Time. In *Proc. ICCV*, 2011. 2, 3

[62] D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26:756–777, 2004. 20, 37

[63] B. M. Oh, M. Chen, J. Dorsey, and F. Durand. Image-based modeling and photo editing. In *Proc. ACM SIGGRAPH*, 2001. 83

[64] T. Okatani and K. Deguchi. On the wiberg algorithm for matrix factorization in the presence of missing components. *Int. J. Comput. Vision*, 72(3):329–337, 2006. 22

[65] J. Oliensis and R. Hartley. Iterative extensions of the sturm/triggs algorithm: Convergence and nonconvergence. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(12):2217–2233, 2007. 3, 22

[66] Y. I. H. Parish and P. Müller. Procedural modeling of cities. In *Proc. ACM SIGGRAPH*, pages 301–308, 2001. 77, 82

[67] M. Park, K. Brocklehurst, R. T. Collins, and Y. Liu. Deformed lattice

detection in real-world images using mean-shift belief propagation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2009. 5, 54, 57, 58, 59, 71, 72, 73, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131

[68] M. Park, K. Brocklehurst, R. T. Collins, and Y. Liu. Translation-symmetry-based perceptual grouping with applications to urban scenes. In *Proc. ACCV (3)*, pages 329–342, 2010. 54

[69] M. Park, S. Lee, P.-c. Chen, S. Kashyap, A. A. Butt, and Y. Liu. Performance evaluation of state-of-the-art discrete symmetry detection algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2008. 5

[70] M. Pauly, N. J. Mitra, J. Wallner, H. Pottmann, and L. J. Guibas. Discovering structural regularity in 3d geometry. *ACM Trans. on Graph. (Proc. of SIGGRAPH)*, 2008. 5, 55, 57

[71] C. J. Poelman and T. Kanade. A paraperspective factorization method for shape and motion recovery. In *Proc. ECCV (2)*, pages 97–108, 1994. 3

[72] M. Pollefeys, D. Nistér, J. M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S. J. Kim, P. Merrell, C. Salmi, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewénius, R. Yang, G. Welch, and H. Towles. Detailed real-time urban 3d reconstruction from video. *Int. J. Comput. Vision*, 78:143–167, 2008. 5, 9, 84

[73] A. Ranganathan, E. Menegatti, and F. Dellaert. Bayesian inference in the space of topological maps. *IEEE Transactions on Robotics*, 22:92–107, 2006. 32

[74] R. Roberts, S. N. Sinha, R. Szeliski, and D. Steedly. Structure from motion for scenes with large duplicate structures. In *Proc. CVPR*, 2011. 29, 32, 37, 40, 46, 50

[75] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *Proc. ECCV*, pages 430–443, 2006. 2

[76] C. A. Rothwell, D. A. Forsyth, A. Zisserman, and J. L. Mundy. Extracting projective structure from single perspective views of 3d point sets. In *Proc. ICCV*, 1993. 81

[77] F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets, or "how do i organize my holiday snaps?". In *Proc. ECCV*, pages 414–431, 2002. 31

[78] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proc. CVPR*, pages 519–528, 2006. 2

[79] V. Shiv Naga Prasad and L. S. Davis. Detecting rotational symmetries. In *Proc. ICCV*, 2005. 5, 56

[80] S. Sinha, D. Steedly, and R. Szeliski. A multi-staged linear approach to structure from motion. In *In RMLE-ECCV workshop*, 2010. 24, 31

[81] S. N. Sinha, D. Steedly, R. Szeliski, M. Agrawala, and M. Pollefeys. Interactive 3d architectural modeling from unordered photo collections. *ACM Trans. on Graph. (Proc. of SIGGRAPH Asia)*, pages 1–10, 2008. 77, 84, 93, 94, 104, 106

[82] N. Snavely, S. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3d. *ACM Trans. on Graph.*, 25:835–846, 2006. 25

[83] N. Snavely, S. Seitz, and R. Szeliski. Modeling the world from internet photo collections. *Int. J. Comput. Vision*, 80:189–210, 2008. 2, 25, 28, 29, 31, 38, 46, 47, 48, 49

[84] V. Starovoitov, S. Jeong, and R. Park. Texture periodicity detection: Fea-

tures, properties, and comparisons. *IEEE Trans. Systems, Man and Cybernetics, Part A*, 28(6):839–848, 1998. 56

[85] P. F. Sturm and B. Triggs. A factorization based algorithm for multi-image projective structure and motion. In *Proc. ECCV (2)*, pages 709–720, 1996. 3, 22

[86] Y. W. Tai, M. S. Brown, C. K. Tang, , and H. Y. Shum. Texture amendment: Reducing texture distrotion in constrained parameterization. *ACM Trans. on Graph.*, pages 1–6, 2008. 98

[87] S. Thrun and B. Wegbreit. Shape from symmetry. In *Proc. ICCV*, pages 1824–1831, 2005. 57

[88] C. Tomasi. Shape and motion from image streams under orthography: A factorization method. *Int. J. Comput. Vision*, 9:137–154, 1992. 3, 21

[89] B. Triggs. Factorization methods for projective structure and motion. In *Proc. CVPR*, pages 845–851, 1996. 21

[90] B. Triggs, P. Mclauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment - a modern synthesis. *Lecture Notes in Computer Science*, pages 298–375, 2000. 2, 25

[91] A. Van Den Hengel, A. Dick, T. Thormählen, B. Ward, and P. Torr. Videotrace: Rapid interactive scene modelling from video. *ACM Trans. on Graph. (Proc. of SIGGRAPH)*, 2007. 93

[92] J. Wang, X. Tong, S. Lin, M. Pan, C. Wang, H. Bao, B. Guo, and H. Y. Shum. Appearance manifolds for modeling time-variant appearance of materials. *ACM Trans. on Graph. (Proc. of SIGGRAPH)*, pages 754–761, 2006. 99

[93] M. Wilczkowiak, E. Boyer, and P. Strum. Camera calibration and 3d recon-

struction from single images using parallelepipeds. In *Proc. ICCV*, pages 142–148, 2001. 18, 19, 86

[94] M. Wilczkowiak, P. Sturm, and E. Boyer. Using geometric constraints through parallelepipeds for calibration and 3d modeling. *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 194–207, 2005. 83, 85, 86

[95] C. Wu, J.-M. Frahm, and M. Pollefeys. Detecting large repetitive structures with salient boundaries. In *Proc. ECCV*, 2010. 5, 54, 56, 67

[96] J. Xiao, T. Fang, P. Tan, P. Zhao, E. Ofek, and L. Quan. Image-based façade modeling. *ACM Trans. on Graph. (Proc. of SIGGRAPH Asia)*, 27(5):1–10, 2008. 56, 84, 93, 104

[97] J. Xiao, T. Fang, P. Zhao, M. Lhuillier, and L. Quan. Image-based street-side city modeling. *ACM Trans. on Graph. (Proc. of SIGGRAPH Asia)*, 2009. 5, 9, 56, 77, 112

[98] C. Zach, A. Irschara, and H. Bischof. What can missing correspondences tell us about 3d structure and motion? In *Proc. CVPR*, 2008. 29, 32, 34, 35

[99] C. Zach, M. Klopschitz, and M. Pollefeys. Disambiguating visual relations using loop constraints. In *Proc. CVPR*, pages 1426–1433, 2010. 28, 29, 31, 39, 42, 46, 47, 48, 49, 50

[100] L. Zebedin, A. Klaus, B. Gruber-Geymayer, and K. Karner. Towards 3d map generation from digital aerial images. *Journal of Photogrammetry and Remote Sensing*, pages 413–427, 2006. 84

[101] Z. Y. Zhang. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22:1330–1334, 2000. 15

[102] Z. Y. Zhang and H. T. Tsui. 3d reconstruction from a single view of an

object and its image in a plane mirror. In *In Proc. of ICPR*, 1998. 9, 79, 81, 92

[103] P. Zhao and L. Quan. Translation symmetry detection in a fronto-parallel view. In *Proc. CVPR*, pages 1009–1016, 2011. 54, 56, 73

[104] P. Zhao, L. Yang, H. Zhang, and L. Quan. Per-pixel translational symmetry detection, optimization, and segmentation. In *Proc. CVPR*, pages 526–533, 2012. 56

[105] Q. Zheng, A. Sharf, G. Wan, Y. Li, N. J. Mitra, D. Cohen-Or, and B. Chen. Non-local scan consolidation for 3d urban scenes. *ACM Trans. on Graph. (Proc. of SIGGRAPH)*, 2010. 53, 58

[106] Z. Zhengdong, G. Arvind, L. Xiao, , and M. Yi. Tilt: Transform invariant low-rank textures. *Int. J. Comput. Vision*, 99(1):1–24, 2012. 57

[107] Z. Zhengdong, L. Xiao, and M. Yi. Unwrapping low-rank textures on generalized cylindrical surfaces. In *Proc. ICCV*, pages 1347–1354, 2011. 57