FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Reconstruction of 3D Models from Images and Point Clouds with Shape Primitives

DISSERTATION

zur Erlangung des akademischen Grades

Doktorin der technischen Wissenschaften

eingereicht von

Irene Reisner-Kollmann
Matrikelnummer 0626981

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Prof. Dr. Werner Purgathofer

Diese Dissertation haben begutachtet:

(Univ.Prof. Dr. Werner Purgathofer)

(Ao.Univ.Prof. Dr. Markus Vincze)

Wien, 18.2.2013

(Irene Reisner-Kollmann)

delivered by
EUROGRAPHICS
DIGITAL LIBRARY
www.eg.org
diglib.eg.org

Reconstruction of 3D Models from Images and Point Clouds with Shape Primitives

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktorin der technischen Wissenschaften

by

Irene Reisner-Kollmann
Registration Number 0626981

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Prof. Dr. Werner Purgathofer

The dissertation has been reviewed by:

(Univ.Prof. Dr. Werner Purgathofer)

(Ao.Univ.Prof. Dr. Markus Vincze)

Wien, 18.2.2013

(Irene Reisner-Kollmann)

Erklärung zur Verfassung der Arbeit

Irene Reisner-Kollmann
Albertgasse 45/2/3, 1080 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum)	(Unterschrift Verfasserin)

Acknowledgements

I would like to thank Werner Purgathofer for supervising my PhD thesis and his support and advice during my work. I also thank Markus Vincze for his review of this thesis as second supervisor.

I want to express my gratitude to my colleagues at the VRVis research center. I thank Stefan Maierhofer and Anton Fuhrmann for supervising my doctoral research at VRVis. Further, I thank the whole Aardvark team, which was a great pleasure to work with. A special thank goes to the co-workers of my publications, which have made major contributions to my work: Christian Luksch, Andreas Reichinger, and Michael Schwärzler.

An important factor for the success of this work was my participation in the doctoral college for *Computational Perception*. I want to thank all professors and students, especially Margrit Gelautz and Matthias Bernhard, for valuable discussions and suggestions.

Finally, I want to thank Harald and my whole family for their support during my studies.

Abstract

3D models are widely used in different applications, including computer games, planning software, applications for training and simulation, and virtual city maps. For many of these applications it is necessary or at least advantageous, if the virtual 3D models are based on real world scenes and objects. Manual modeling is reserved for experts as it requires extensive skills. For this reason, it is necessary to provide automatic or semi-automatic, easy-to-use techniques for reconstructing 3D objects.

In this thesis we present methods for reconstructing 3D models of man-made scenes. These scenes can often be approximated with a set of geometric primitives, like planes or cylinders. Using geometric primitives leads to light-weight, low-poly 3D models, which are beneficial for efficient storage and post-processing.

The applicability of reconstruction algorithms highly depends on the existing input data, the characteristics of the captured objects, and the desired properties of the reconstructed 3D model. For this reason, we present three algorithms that use different input data. It is possible to reconstruct 3D models from just a few photographs or to use a dense point cloud as input. Furthermore, we present techniques to combine information from both, images and point clouds.

The image-based reconstruction method is especially designed for environments with homogenous and reflective surfaces where it is difficult to acquire reliable point sets. Therefore we use an interactive application which requires user input. Shape primitives are fit to user-defined segmentations in two or more images.

Our point-based algorithms, on the other hand, provide fully automatic reconstructions. Nevertheless, the automatic computations can be enhanced by manual user inputs for generating improved results. The first point-based algorithm is specialized on reconstructing 3D models of buildings and uses unstructured point clouds as input. The point cloud is segmented into planar regions and converted into 3D geometry.

The second point-based algorithm additionally supports the reconstruction of interior scenes. While unstructured point clouds are supported as well, this algorithm specifically exploits the redundancy and visibility information provided by a set of range images. The data is automatically segmented into geometric primitives. Then the shape boundaries are extracted either automatically or interactively.

Kurzfassung

3D-Modelle finden immer breitere Anwendung in verschiedensten Applikationen, etwa in Computerspielen, Planungssoftware, Applikationen für Training und Simulation, oder in virtuellen Stadtplänen. Für viele dieser Anwendungen ist es notwendig oder zumindest vorteilhaft, wenn die 3D Modelle einer echten Szene entsprechen. Da die manuelle Modellierung echter Objekte nur Spezialisten vorbehalten ist, ist es notwendig automatische oder semi-automatische, einfach zu bedienende Verfahren zur Rekonstruktion von Objekten zu entwickeln.

Die vorliegende Arbeit präsentiert Methoden zur Rekonstruktion von 3D Modellen, die speziell für künstliche, von Menschen geschaffenen Umgebungen entwickelt wurden. Diese Objekte können meist durch einfache geometrische Primitive, wie z.B. Ebenen oder Zylinder, approximiert werden. Der Vorteil dieser geometrischen Primitive ist, dass die rekonstruierten Modelle sehr einfach gehalten werden und dadurch eine effiziente Speicherung und Weiterverarbeitung ermöglichen.

Die Anwendbarkeit von Rekonstruktionsalgorithmen ist stark abhängig von den vorhandenen Eingabedaten, der Beschaffenheit der aufgenommenen Objekte, sowie den gewünschten Eigenschaften des erstellten 3D Modells. Aus diesem Grund präsentieren wir drei verschiedene Algorithmen, welche unterschiedliche Eingabedaten verwenden. Es ist möglich, Objekte aufgrund von mehreren registrierten Bildern der Szene oder anhand einer Punktwolke zu rekonstruieren. Weiters präsentieren wir Methoden um die Information aus beiden Datentypen zu kombinieren.

Unsere bildbasierte Rekonstruktionsmethode ist speziell für Umgebungen mit homogenen und glänzenden Oberflächen ausgerichtet, bei denen die Aufnahme von zuverlässigen 3D Punktdaten nur schwer möglich ist. Aus diesem Grund entwickelten wir eine interaktive Applikation, bei der Benutzereingaben notwendig sind. Geometrische Primitive werden anhand von benutzerdefinierten Segmentierung in zwei oder mehr Bildern in die Szene eingepasst.

Bei den punktbasierten Rekonstruktionsmethoden ist es hingegen möglich, 3D-Modelle vollautomatisch zu rekonstruieren. Jedoch besteht auch hier die Möglichkeit, die Ergebnisse mit Benutzereingaben zu verbessern. Der erste punktbasierte Algorithmus ist auf die Rekonstruktion von Gebäuden spezialisiert und verwendet unstrukturierte Punktwolken als Eingabe. Die Punktwolke wird in planare Segmente unterteilt und in 3D-Geometrie umgewandelt.

Der zweite Algorithmus ist zusätzlich für die Rekonstruktion von Innenräumen geeignet. Obwohl auch hier unstrukturierte Punktwolken unterstützt werden, wertet dieser Algorithmus spezielle Eigenschaften von Tiefenbildern aus, wie z.B. Redundanz und Sichtbarkeitsinformationen. Die automatische Segmentierung detektiert verschiedene geometrische Primitive. Anschließend werden die Begrenzungen der Primitive automatisch oder interaktiv extrahiert.

Contents

1 **Introduction** **1**
- 1.1 Problem Statement 2
- 1.2 Contributions 2

2 **Related Work** **5**
- 2.1 Data Acquisition 5
  - 2.1.1 Photogrammetry 6
  - 2.1.2 Laser Scans 9
  - 2.1.3 Range Images 10
- 2.2 General Concepts 12
- 2.3 Point Cloud Reconstruction 16
  - 2.3.1 Interactive Approaches 16
  - 2.3.2 Automatic systems 17
- 2.4 Image-based Modeling 20
  - 2.4.1 Interactive Approaches 20
  - 2.4.2 Automatic reconstruction 23

3 **Image-Based Modeling of Industrial Environments** **27**
- 3.1 Introduction 27
  - 3.1.1 Problem statement 28
- 3.2 Camera Orientation 29
- 3.3 Model Fitting 31
  - 3.3.1 Models 31
  - 3.3.2 Image Constraints 32
  - 3.3.3 Relational Constraints 34
  - 3.3.4 Initialization and Optimization 35
  - 3.3.5 Reconstruction of Tubes 35
- 3.4 Results 37
- 3.5 Conclusions 38

4 **Automatic Building Reconstruction from Point Clouds** **41**
- 4.1 Introduction 41
  - 4.1.1 Input Data 42

4.2 Point Cloud Segmentation . . . 43
4.2.1 Axes Search . . . 43
4.2.2 Planar Segmentation . . . 44
4.2.3 Point Evaluation . . . 44
4.3 Polygonization . . . 45
4.3.1 Point Rasterization . . . 45
4.3.2 Image Projection . . . 46
4.3.3 Edge Detection . . . 46
4.3.4 Grid Filling . . . 47
4.3.5 Combination and Texturing of Polygons . . . 47
4.4 Results . . . 49
4.5 Conclusions . . . 49

5 Interactive Reconstruction from RGB-D Images 51
5.1 Segmentation . . . 51
5.1.1 Graph setup . . . 53
5.1.2 Global segmentation . . . 54
5.1.3 Frame segmentation . . . 56
5.1.4 Results . . . 57
5.2 Boundary Optimization . . . 58
5.2.1 Constraints . . . 60
5.2.2 Rasterization artifacts . . . 60
5.3 Constraints . . . 62
5.3.1 Point Cloud . . . 62
5.3.2 Depth Maps . . . 63
5.3.3 Shape Intersections . . . 65
5.3.4 Images . . . 65
5.3.5 Coplanarity . . . 66
5.3.6 2D Shapes . . . 67
5.3.7 User Input . . . 67
5.4 Evaluation . . . 69
5.4.1 Synthetic datasets . . . 69
5.4.2 Real-world datasets . . . 70
5.5 Conclusions . . . 74

6 Conclusions and Outlook 75
6.1 Conclusions . . . 75
6.2 Outlook . . . 76

Bibliography 79

A Curriculum Vitae 93

CHAPTER 1

Introduction

Reconstructing virtual 3D models of real-world scenes is an exciting topic with a large variety of possible applications. In recent years, reconstruction has gained a lot of attention due to new capturing methods. Photogrammetric tools as well as consumer depth cameras enable an intuitive and cost-efficient way of capturing scenes as point clouds. However, for many applications a point cloud is not sufficient and it is necessary to obtain a polygonal representation.

3D reconstruction is an important tool for maintaining cultural heritage. In some cases, where physical monuments are endangered, digital preservation might be the only possibility. Having 3D models of important cultural heritage allows the presentation to more people than would be possible at the original site. Furthermore, non-invasive reconstruction methods enable the detection of small details which would not be possible otherwise.

Having accurate 3D models of buildings provides advantages for many applications. 3D information is a valuable input for planning modifications of buildings, or for planning the placement of furniture in interior rooms. 3D models provide information for security applications, e.g. fall detection of elderly people. For robots it is also essential to have 3D information in order to freely navigate and execute several tasks. Another field of applications is virtual training, where people train specific tasks such as fire fighting. Having realistic 3D views of scenes makes such training setups much more immersive and successful.

Digital map services, such as Google maps or Bing maps, have gained great success with including 3D models into their virtual city maps. Reconstructions of large city models also improve navigation systems by visualizing the surrounding environment in 3D.

Using real-world models for computer games can increase the identification of the user with the virtual world. There are plenty of possibilities, e.g. using models of well-known landmarks or of whole cities. It is also imaginable that users provide data from surrounding environments. While it is not always necessary to have an exact copy of a real scene, reconstructions can provide information about the style and underlying structures of buildings or cities. This information can be used for automatically generating new virtual worlds with procedural modeling techniques.

The large variety of applications produces different requirements on reconstruction systems. For some reconstructions it is necessary to have very exact measurements. Other applications might tolerate approximations in order to generate complete models. The reconstruction techniques have to be easy-to-use in order to be accessible for non-experts and thus being used for a wide range of applications. This means, that the algorithms are either fully automatic or that they have an intuitive user interface.

1.1 Problem Statement

The primary goal of this thesis is to provide 3D reconstruction techniques for buildings, interior scenes, and other man-made shapes. The scenes should be reconstructed by simple low-poly meshes in order to facilitate further editing and rendering. A wide range of man-made scenes can be approximated with piecewise planar models or with a small set of geometric primitives.

A reconstruction algorithm heavily depends on the data that is available from the scene. Depending on the acquisition procedure, it is possible to have multiple images, sparse or dense point clouds, or a mixture of images and point clouds. Besides specialized algorithms for individual input data, it is useful to combine information from all possible input sources.

Several challenges arise for reconstructing reasonable models from images or point clouds. Usually, not all parts of a scene can be captured. Especially in interior scenes, some areas are hard to access because they are occluded by other objects. Texture-poor surfaces and repeating structures form a problem for photogrammetric tools, while specular surfaces are also problematic for active sensors such as laser scanners. An important task is to find efficient and reliable ways for filling in the missing data, e.g. by surrounding measurements, prior information about the reconstructed scene, or by employing user input.

1.2 Contributions

The contributions of this thesis are several solutions for reconstructing simple 3D models from point clouds and images. We focus on reconstructing buildings, interior scenes, and other man-made shapes. Therefore we generate either piecewise planar models or we use a set of simple shape primitives that can be tesselated with any desired accuracy. When it is possible, we combine information from different input sources such as point clouds and images. Due to different input data and scene characteristics, the presented techniques use very different algorithms. Nevertheless, the overall structure is similar for all techniques, as can be seen in Figure 1.1. First the input data is segmented into shape primitives. Then the shape parameters and optionally the shape boundaries are optimized for generating a 3D mesh. Finally, if images of the scene are available, textures can be projected onto the 3D model.

In Chapter 3 we present an interactive algorithm for reconstructing interior scenes solely from multiple images [112]. The algorithm is specialized for industrial environments and therefore has to deal with specular and texture-poor surfaces. It is necessary to include user input as automatic matching and segmentation techniques are unreliable for such difficult scenes. The user can indicate the placement of geometric primitives such as boxes or superquadrics in multiple images. The accurate parameters of the geometric primitives are computed automatically.

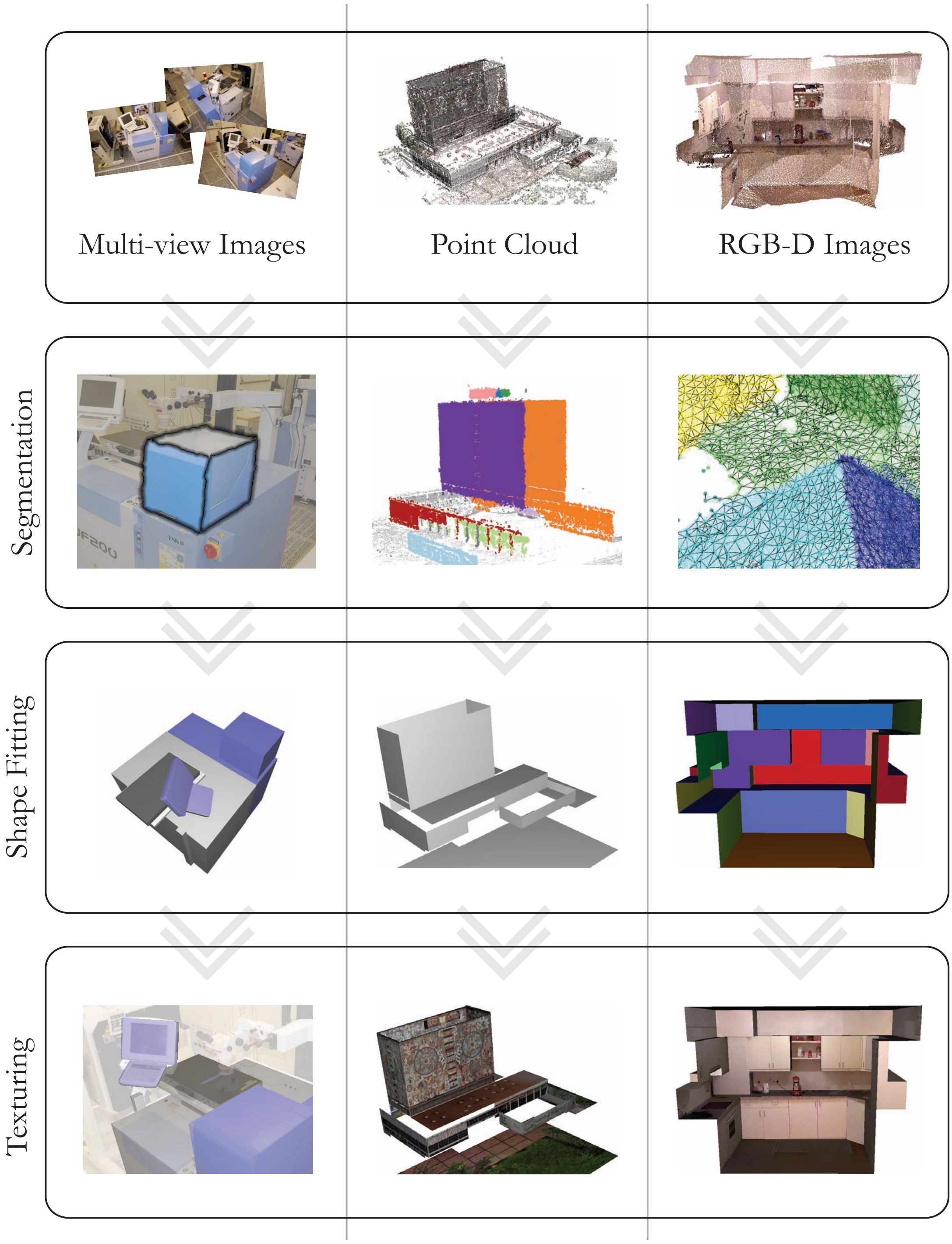


Figure 1.1: Overview of proposed algorithms for multi-view images, point clouds, and RGB-D images. The input data is segmented into shape primitives which are used for generating a 3D model. Finally the models are textured by input images.

In Chapter 4 we focus on reconstructing piecewise planar models of buildings from point sets [113]. For outdoor scenes like buildings it is easily possible to acquire dense point sets, either from images or with a laser scan, which enables us to apply a fully automatic reconstruction algorithm. In this approach, we first determine dominant surface normals in the scene and subsequently detect planes. The plane detection prefers nice angles such as $90°$, $60°$, or $45°$, which is a reasonable assumption for the majority of buildings. The boundaries of the planes are computed either from the point cloud itself, or by additionally including information from images. The images provide more accurate information and allow to exactly align boundaries with strong image edges.

Combining information from images and point clouds is also the goal in Chapter 5. We focus on the usage of RGB-D images but most parts of the algorithm can be used for other input sources as well. The algorithm first starts with a novel segmentation technique [115] for multiple RGB-D images which efficiently handles the large amount of redundant data. The segmentation divides the point set into multiple primitive shapes. Then we present a new optimization framework for detecting reasonable boundaries of the primitive shapes [116]. In this optimization framework, it is possible to include input information from images, point clouds, and depth maps. Additionally, high level constraints can be defined such as intersections, coplanar boundary lines or 2D shapes. While the algorithm can be run automatically, it is also possible to include user-defined constraints.

CHAPTER 2

Related Work

This chapter provides a comprehensive overview of reconstruction algorithms that generate 3D models from point clouds or images. We focus on man-made scenes such as buildings, interior scenes, or engineering objects. The reconstruction should deliver a 3D model which is as simple as possible for a desired level of detail. Furthermore, it is desirable to get information about the underlying structure of a scene for enabling further editing. These goals differ from reconstructing terrains or other free-form surfaces, where it is often sufficient to smoothly interpolate an input point cloud.

We first describe the possibilities for data acquisition on-site, which are photogrammetry, laser scans, and range cameras. The acquisition techniques either generate a set of registered photographs or a sparse or dense point set. The following section presents some general concepts, which are often used for reconstructing man-made objects and scenes. Finally, we present a wide range of up-to-date algorithms, which are divided into image-based methods and methods using point clouds as input. Additionally, we divide the algorithms into automatic and interactive techniques.

It has to be stated, that these separations are not always clearly defined. There are algorithms mainly dealing with point clouds, but which can be enhanced with information from images. Similarly, many image-based techniques require a sparse or even a dense set of 3D points reconstructed from the input images. There are also different stages of required user input, ranging from fully automatic reconstruction over semi-automatic approaches to completely manual modeling. We have classified the algorithms based on their main contribution. This means that some methods are located in the section for automatic reconstruction, although they support optional user input or pre-processing tasks include manual input.

2.1 Data Acquisition

In this section we describe different possibilities for measuring real-world scenes, namely photogrammetry, laser scanning, and range cameras. These techniques generate multiple registered

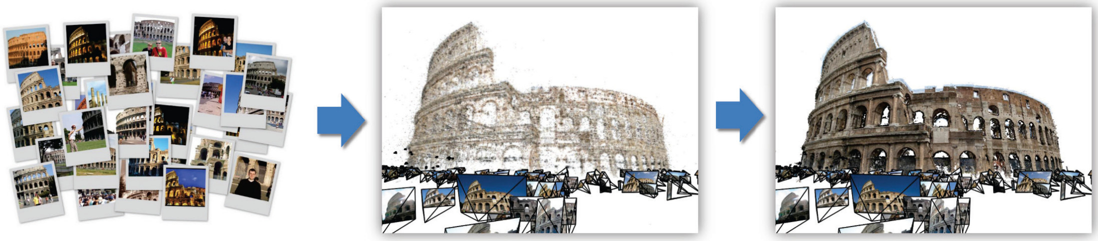


Figure 2.1: Pipeline of photogrammetric reconstruction: Multiple images (left) are registered by structure-from-motion (middle) and dense 3D geometry is generated by multiview stereo (right) [1]. ©IEEE.

images and/or a point cloud of the measured scene. Photogrammetric reconstruction estimates the viewpoints of multiple input photographs and additionally creates a sparse or dense point set of the scene. Laser scanners and range cameras directly generate dense point sets of a scene. Some range cameras additionally provide grayscale or color images.

Choosing a suitable acquisition technique depends on many parameters, including scale and type of the scene as well as available funding. While photogrammetric reconstruction provides a cost-efficient method for different scales, laser scans and active range cameras provide better results for homogeneous surfaces. Extensive comparisons between photogrammetric reconstruction and laser scans are provided by Leberl et al. [79].

2.1.1 Photogrammetry

Photogrammetric reconstruction restores three-dimensional information from one or more images. In this thesis, we mainly focus on the reconstruction from multiple images, which is called stereo vision. Epipolar geometry describes the properties and geometric relations between a 3D scene and its projection onto two or more 2D images. Detailed information about cameras and epipolar geometry can be found in the textbooks by Hartley and Zisserman [53] and by Szeliski [142].

Figure 2.1 shows the typical workflow for photogrammetric reconstruction. The first step of photogrammetric reconstruction involves the registration of all input images. This procedure is called structure-from-motion and includes the computation of intrinsic and extrinsic camera parameters. For registered images, it is possible to compute three-dimensional positions from two or more corresponding image points. Multi-view stereo algorithms use these conditions and compute dense point clouds or triangulated meshes from the input scene.

Structure-from-motion Structure-from-motion (SfM) deals with the registration of multiple images to each other, i.e. the computation of exterior parameters of the cameras. These exterior parameters are the camera position and the camera orientation. Interior camera parameters (focal length, principal point, distortion parameters) can either be computed in a preprocessing calibration step for a specific camera, or their computation is included in the SfM algorithm (self-calibration).

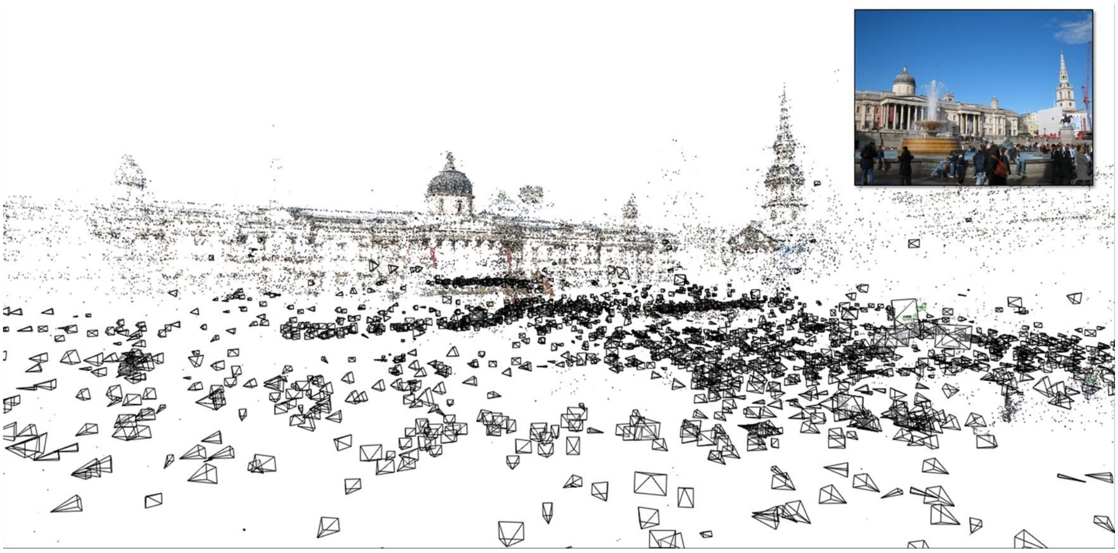


Figure 2.2: A large set of input images has been used for sparse reconstruction by a structure-from-motion algorithm [137]. ©IEEE.

Feature-based approaches are widely used to register images. Local interest points [88] are detected in each image and feature descriptors are used to find corresponding points in other images [122, 135]. Feature matching and estimating the camera parameters is done for pairs of images. For each image pair, at least five point correspondences in case of calibrated images or eight point correspondences in case of self-calibration are necessary for computing the relative position and orientation [53].

Feature correspondences are merged to tracks, which denote corresponding points across multiple views, in order to connect multiple images. Finally, bundle adjustment is used for computing camera parameters and globally consistent 3D positions for the tracks. A nonlinear-optimization, usually Levenberg-Marquardt [87], simultaneously optimizes the 3D positions of all tracks and the camera parameters of all input images, such that the 3D positions are projected close to their initial 2D measurements. The resulting 3D point cloud provides a sparse representation of the scene. Therefore structure-from-motion is also called sparse reconstruction. Figure 2.2 shows an example of a sparsely reconstructed scene together with the camera poses.

Recently, structure-from-motion has been applied to large sets of photos, gathered from internet community photo collections such as Flickr [1, 2, 137]. The main challenge for these approaches is the large amount of data. Fast solutions are achieved by reducing the search space for corresponding images [102] and by exploiting modern graphics hardware [38]. Photo tourism [135] provides an intuitive visualization and scene navigation for large sets of registered images, which has been adapted for Microsoft Photosynth.

Multiview Stereo Structure-from-motion produces a sparse reconstruction of a scene, where the images are registered and only a few 3D points are generated. The subsequent process,

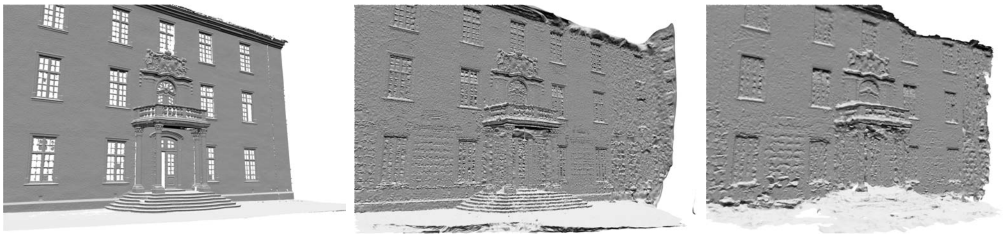


Figure 2.3: Applying multi-view stereo algorithms on the benchmark data *entry-P10* provided by Strecha et al. [140]. The pictures show from left to right: ground-truth, result by Furukawa et al. [43] and by Hiep et al. [58].

multi-view stereo, computes a dense reconstruction from the registered images. Multi-view stereo algorithms try to establish the 3D position for all pixels in the input images. The output is either a triangle mesh or a dense set of points or oriented surface patches. The reconstructed meshes generally have a very large number of triangles.

Seitz et al. [128] gave a review about different multi-view stereo algorithms and provided a benchmark for evaluating them. The datasets of the benchmark as well as the results of many multi-view stereo algorithms are available at `http://vision.middlebury.edu/mview/`. Another benchmark, provided by Strecha et al. [140], focuses on the reconstruction of outdoor scenes and buildings (`http://cvlab.epfl.ch/~strecha/multiview/denseMVS.html`). Figure 2.3 shows a comparison between different multi-view stereo algorithms applied to multiple images of a building.

Volumetric methods are mainly suitable for rather small objects that have been captured from all sides. Space carving and voxel coloring methods use a 3D grid of voxels and remove voxels that are not photo-consistent, i.e. their color does not appear similar in all images. Level set methods [110] and volumetric graph cuts [74] generate a surface by dividing the 3D space into inside or outside areas, which is achieved by minimizing a cost function. The cost function usually optimizes photo-consistency and use some regularization term for generating smooth surfaces.

The patch-based multi-view stereo algorithm (PMVS), presented by Furukawa et al. [43,45], creates a dense set of oriented rectangular patches covering the surface. Starting with the sparse reconstruction from SfM, the set of corresponding points is iteratively expanded to nearby pixels. A filtering step is applied to remove incorrect matches which violate visibility constraints. The increased number of point correspondences can be used for refining the camera parameters with an additional bundle adjustment step [44].

Hiep et al. [58] first compute a dense set of points from the input images by detecting additional interest points and matching them with normalized cross-correlation. The point set is converted into a visibility-consistent mesh with Delaunay triangulation and graph cuts. The initial mesh is refined with a variational approach, that optimizes photo-consistency, regularization and adaptive resolution.

Another approach for computing dense structures is to compute depth maps for each input image and then combine the resulting depth maps with each other. The depth maps can be computed pairwise or multiple images are used for one reference image. Goesele et al. [47] apply stereo matching to each input view with a window-based approach, where corresponding pixels are determined by the normalized cross-correlation. The depth maps are combined into a weighted signed distance volume and the surface is extracted at the zero level [30].

A multi-view stereo algorithm specialized on urban scenes has been presented by Pollefeys et al. [109]. Video streams are connected with GPS (global positioning system) and inertia measurements for achieving a fast and accurate registration with 2D feature tracking. Dense reconstruction is obtained by creating depth maps with plane sweeping and fuse multiple depth maps into triangular meshes.

Architectural scenes and especially interior scenes contain large texture-poor areas, for which feature matching is not possible and thus standard multi-view stereo algorithms do not provide accurate 3D information. Furukawa et al. [41] propose an algorithm based on the Manhattan world assumption. First, a dense point set is generated for textured areas with PMVS [43]. The point set is used for extracting three dominant orthogonal directions and generating hypothesis planes along these directions. Depth maps are created for each input image by assigning one of the plane hypotheses to each pixel. The assignment is formulated as a Markov Random Field and solved with graph cuts. The smoothness term is enhanced by identifying dominant edges in the input images, which are a good position for plane intersections.

Sinha et al. [132] propose a similar multi-view stereo approach based on plane sweeping. They first detect dominant directions by vanishing points [133], which are not restricted to just three directions of a Manhattan world. Sparse 3D line segments along dominant directions are used together with 3D points obtained by SfM for extracting candidate planes. Graph-cut-based energy minimization assigns each pixel to one of the candidate planes. The data term is based on photometric consistency between input images, proximity of nearby 3D points and lines, and visibility constraints. The smoothness term prefers label changes along line segments and vanishing directions.

2.1.2 Laser Scans

A widely-used method for acquiring digital models from real-world scenes is using laser scanners, also referred to as LiDAR (light detection and radar). Laser scanners produce dense unstructured point clouds of a scene with triangulation or with a time-of-flight (ToF) technique. Time-of-flight scanners emit a pulse of light and measure the amount of time before the reflected light is seen again by the detector. Triangulation scanners also emit light, but compute the distance to the measured surface by analyzing where the reflected light appears in the camera's field of view.

LiDAR data is divided into two main categories based on the point of acquisition: Terrestrial LiDAR is acquired by ground-based devices and is mainly used for reconstructing the facades of buildings. Airborne LiDAR data is captured from the air and mainly used for producing building footprints, reconstructing roofs and 2.5D building models [152]. Hybrid approaches combine data from both types of scans [13, 51], and also incorporate photographs [40, 83].

Extensive information about airborne laser scanning is given in the textbook by Campbell [18]. Pfeifer and Briese [107] describe geometrical aspects of laser scanning and provide

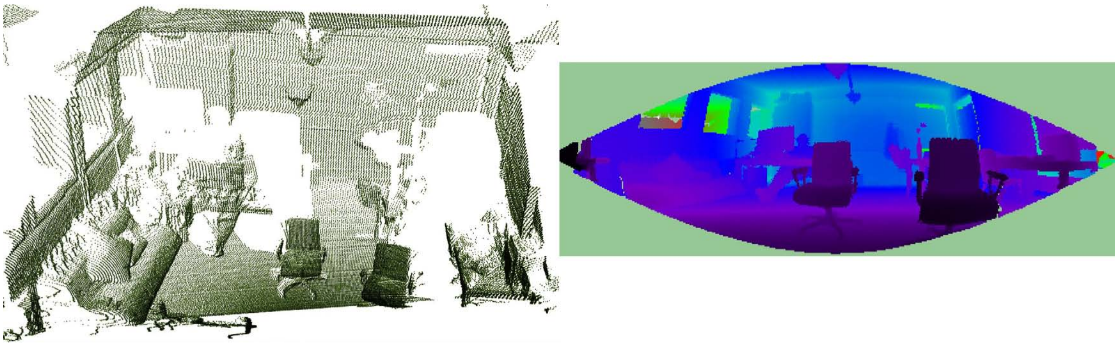


Figure 2.4: A given 3D point cloud (left) is visualized as range image (right) with the Point Cloud Library [120]. ©IEEE.

an overview on several systems and applications. Popular projects in the field of laser scanning are the Michelangelo project [81], the acquisition of the Plastico di Roma antica [50], and the Bayon Digital Archive Project [8]. The Point Cloud Library (PCL) [120] provides many operations on point clouds, including filtering, feature estimation, segmentation, or visualization (see Figure 2.4).

2.1.3 Range Images

Range cameras provide a simple way for capturing real-world scenes in the form of depth maps. They are faster in acquisition than laser scanners and thus they are able to capture moving objects. On the other hand, they usually have a lower resolution, accuracy, and distance measurement range than laser scanners [108]. Microsoft Kinect provides a very cost-efficient way for digitizing 3D scenes, which additionally includes color information (see Figure 2.5). The combination of a depth and a color image from the same viewpoint is called RGB-D image. According to their measurement techniques, range cameras can be divided into time-of-flight cameras, structured light sensors, and passive stereo cameras.

Time-of-flight (ToF) cameras appear in two main variations. One method computes the distance based on the direct measurement of the runtime of a travelled light pulse. The other method uses amplitude modulated light and obtains distance information by measuring the phase difference between the emitted and the received signal. Distortions caused by object reflectivity and internal light scattering have to be handled by calibration [68] in order to obtain precise measurements. ToF sensors have a low spatial resolution compared to other range cameras. Extensive statistics about depth measurements as well as calibration techniques are available for specific ToF cameras, e.g. SwissRanger SR-2 and SR-3000 [67], SR-4000 [25], and Photonic Mixer Device (PMD) [85]. A comparitive evaluation between SwissRanger SR-4000, Fotonic B70 and Microsoft Kinect has been conducted by Stoyanov et al. [139].

Structured light sensors project a known pattern onto a scene. When the scene is captured, the points of the pattern are localized, and it is possible to compute their depth value depending on their transformation to the emitting position. Microsoft Kinect uses an infrared projector and camera, which has the benefit that the scene is not affected by the structured light for the ac-

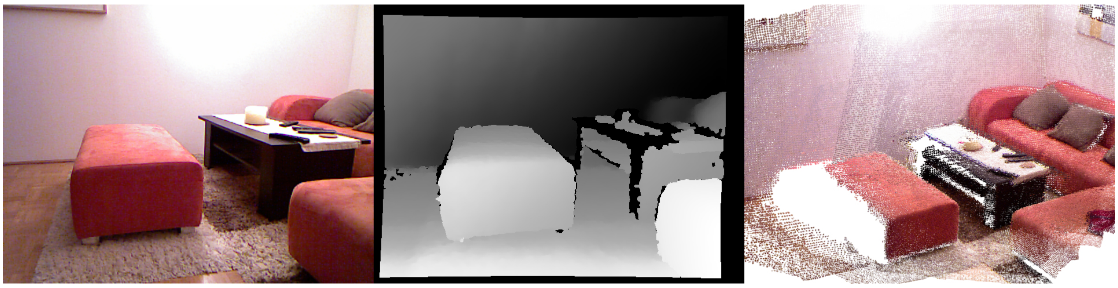


Figure 2.5: Capturing an interior scene with Microsoft Kinect (from left to right): Color image, depth image, 3D point cloud from multiple Kinect frames.

companying color camera. Khoshelham [70] and Smisek et al. [134] analyze the reconstruction technique of Microsoft Kinect and provide information about depth measurement resolution and error properties.

Passive stereo cameras capture a scene with two or more image sensors. The depth values are computed by triangulating corresponding points between the images, which is highly related to the algorithms used for structure-from-motion described in Section 2.1.1. An evaluation of dense two-frame stereo algorithms is provided by Scharstein and Szeliski [123]. Textureless regions and repeated patterns lead to missing or wrong data, because correspondences between images cannot be established. Combining a stereo camera with a ToF-sensor leads to improved results, because they have complementary error characteristics [76, 165].

Multiple range images have to be registered in order to compute the camera viewpoints and to generate a combined point set of a larger scene. This task is called SLAM, simultaneous localization and mapping, when it is applied by robots to explore an unknown environment. Henry et al. [33, 56, 57] first perform a pairwise alignment with SIFT features from the color images [88] and RANSAC [35]. Three corresponding 3D points are randomly selected for computing the relative pose which is tested by all corresponding points. The estimated pose which produced the highest number of inliers is accepted. The estimated pose is refined with Iterative Closest Points (ICP) [119], which improves the alignment of two point sets by iteratively associating nearest points between the point sets. Due to the pairwise registration, errors in pose estimation are accumulated. This problem can be handled with loop closure, when an earlier observed area is captured again. Nearby frames are registered to each other and the registration is globally optimized with the TORO minimization framework [48].

Kinect Fusion [61, 100] is a system for tracking a range camera and fusing the range images in real-time (see Figure 2.6). The registered range images are integrated in a volumetric surface representation as truncated signed distance functions (TSDF). For a new viewpoint, a virtual range image is generated from the existing 3D data with raycasting. New range images are aligned with the raycasted depth image of the previous viewpoint using ICP. The main advantage of this approach is, that a new frame can be efficiently registered to the complete 3D data, instead of just using the previously aligned frame.

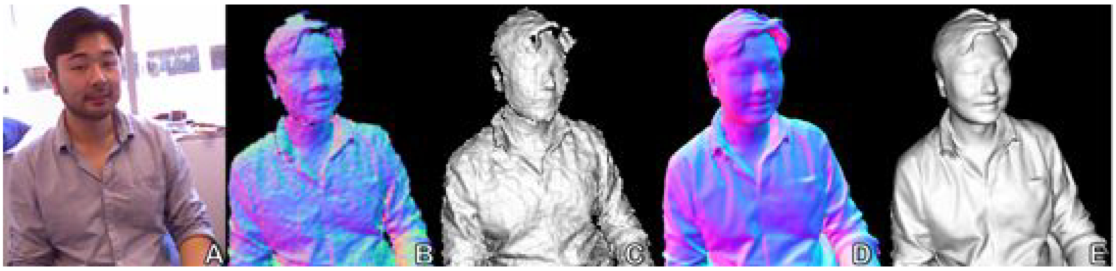


Figure 2.6: KinectFusion: A) RGB image. B anc C) Normals and 3D Mesh from single depth map. D and E) 3D model generated from fused TSDF [61]. ©ACM.

2.2 General Concepts

In this section we describe some concepts and assumptions that are often used for reconstructing building interiors and exteriors as well as other man-made shapes. These objects often contain regular structures, which can be exploited for robustly reconstructing noisy or incomplete data scans. Analyzing the underlying structure of a scene is also useful for subsequent manipulations. For example, the arrangement of windows and other facade objects can be automatically adapted while a building is resized. Such operations are hardly possible when a scene is represented with a large set of triangles.

The amount of prior information usually forms a trade-off between robustness and generality. Using appropriate priors allows to reconstruct reasonable 3D models also in case of large artifacts in the input data. On the other hand, applying very strict assumptions reduces the number of possible scene contents for a reconstruction algorithm. Table 2.1 provides an overview of important concepts that are used by different algorithms and which will be presented in detail in the following part of this section.

Reconstruction techniques can be divided into volumetric and surface-oriented approaches. Volumetric approaches assume that objects can be fully reconstructed and create watertight models without any holes. They partition the 3D space into exterior and interior areas, which is often accomplished by labeling a 3D voxel grid. Surface-oriented reconstruction techniques generate only the surface of an object. They are able to partially reconstruct objects, which is useful if large parts are missing.

The desired level of detail is an important parameter for reconstruction. For example, windows of a building may be interpreted as important structures or as insignificant perturbations depending on the scale of interest. In many algorithms, the level of detail has to be chosen by the user. An important factor for a reasonable value is the quality of the input data, including resolution, noise, and amount of outliers.

For direct reconstruction of point clouds a smoothness assumption is often used, where the empty space between points is interpolated. A large variety of algorithms is available, e.g. radial basis functions [19], moving least squares (MLS) [80], poisson surface reconstruction [69], or the locally optimal projection (LOP) operator [60, 86]. Special algorithms have been proposed for retaining sharp features [6, 37]. These approaches are usually not directly applicable to large scenes captured in the wild, which are highly polluted by noise and contain large areas of missing

Table 2.1: General concepts used by algorithms presented in Sections 2.3 and 2.4. Note that symmetry also includes repetition. Some methods using the manhattan world assumption are also able to reconstruct additional surfaces in other directions.

	Images	Points	User	Manhattan	Planes	Shapes	Symmetry	Volume
Debevec et al. 1996 [31]	+		+			+		
Poulin et al. 1998 [111]	+		+		+			
Robertson and Cipolla 2000 [118]	+		+	+	+			
Werner and Zisserman 2002 [155]	+			+	+	+		
Schindler and Bauer 2003 [124]	+	+			+	+		
Dick et al. 2004 [32]	+					+	+	
El-Hakimet al. 2005 [34]	+		+				+	
Fraundorfer et al. 2006 [39]	+	+			+			
Van den Hengel et al. 2006 [145]	+	+	+			+	+	
Bartoli and Sturm 2007 [10]	+	+			+			
Van den Hengel et al. 2007 [148]	+	+	+					
Müller et al. 2007 [93]	+				+		+	
Chen and Chen 2008 [23]	+	+	+					+
Jenke et al. 2008 [63]		+				+		
Jenke et al. 2008 [64]		+				+		
Rusu et al. 2008 [121]		+			+	+		
Sinha et al. 2008 [133]	+	+	+	+	+			
Xiao et al. 2008 [158]	+	+	+		+		+	
Furukawa et al. 2009 [42]	+	+		+	+			+
Labatut et al. 2009 [77]		+				+		+
Schnabel et al. 2009 [126]		+				+		+
Xiao et al. 2009 [159]	+	+			+		+	
Chauve et al. 2010 [22]		+			+			+
Nan et al. 2010 [97]		+	+	+	+		+	
Vanegas et al. 2010 [149]	+			+		+		+
Kowdle et al. 2011 [75]	+	+	+		+			
Li et al. 2011 [83]	+	+		+	+		+	
Ceylan et al. 2012 [20]	+		+		+		+	
Lafarge et al. 2012 [78]		+				+		+
Wu et al. 2012 [156]		+				+		
Kim et al. 2012 [72]		+				+		
Nan et al. 2012 [98]		+				+		
Shao et al. 2012 [129]		+	+			+		
Simon et al. 2012 [130]	+	+			+		+	
Xiao and Furukawa 2012 [160]	+	+			+			+
Arikan et al. 2013 [4]		+	+		+			

data. In addition, direct reconstruction techniques do not analyze the underlying structure of a scene and generally create a large set of triangles. Nevertheless, some methods can be applied as preprocessing steps such as denoising, outlier removal, thinning, and robust normal estimation.

The Manhattan world assumption states that scenes are made from planar surfaces which are oriented along three mutually orthogonal directions. Many buildings, interiors as well as exteriors, can be approximated to a large extent with this constraint. The Manhattan world assumption has been used for reconstruction of both, point clouds [149, 150] and images [41, 42, 133]. In case of point clouds, the three main directions are usually extracted from surface normals or by fitting planes to the point cloud. In images, main directions are revealed by vanishing points. For a set of registered images, it is possible to extract globally consistent vanishing points [59].

Many algorithms reconstruct scenes with piecewise planar models [7, 10, 39, 77, 132, 133], which provide a reasonable abstraction to many man-made objects. Curved surfaces can be approximated by multiple planar faces. In case of multi-view imagery, a plane induces a homography between corresponding points of two images. If the correct world plane is available, every point in one image can be transferred to its corresponding point in another image. Conversely, homographies and world planes can be established from three or more point correspondences.

Reconstructing scenes only with a small set of geometric primitives can be used for buildings as well as other man-made shapes such as engineering objects. Besides reconstruction the goal is to enrich point sets with abstractions and possibly semantic information. Most algorithms for detecting multiple structures in a point cloud are based on the Hough transform [153], region growing [22, 121], or on random sampling strategies such as RANSAC [35, 77, 127]. Geometric primitives can be directly extracted from the point cloud, e.g. planes, cylinders, cones, or tori [77, 127] or they are generated from a combination of simpler objects, e.g. cuboids are fitted to detected planes [121]. In interactive approaches, the user can place parametric objects in the scene, while the accurate parameters are computed automatically based on the input data. For example, Nan et al. [97] use boxes to reconstruct buildings from point clouds and in the Façade system [31], boxes, prisms and surfaces of revolution are used for image-based modeling.

Geometric primitives are also used by the GlobFit system [82] which automatically detects global relations between multiple primitives (see Figure 2.7). Primitive directions, e.g. plane normals, provide information about parallelism, orthogonality or equal angles across the scene. Further relations include coplanar and coaxial placements as well as equality of object sizes. The parameters of the geometric primitives are optimized to the underlying point set while they account for all detected constraints. Similar relations are used by Nan et al. [97] for positioning boxes in a point cloud.

Symmetry is an important property of a lot of man-made shapes; especially building facades contain a large amount of self-similarity and repetitions [20, 93, 95]. Symmetry can be used to fill in missing data from other parts of the scene, and to denoise point clouds by non-local filtering [164]. Several algorithms have been proposed for detecting different types of symmetry in triangle meshes and point clouds [11, 14, 91, 105] as well as in images [28, 89, 131, 157]. Information about symmetry is also used for improving structure-from-motion algorithms [27, 54] and allows to extract 3D information from single images [65, 131].

Detailed assumptions can be used for reconstructing buildings and facades. The Manhattan

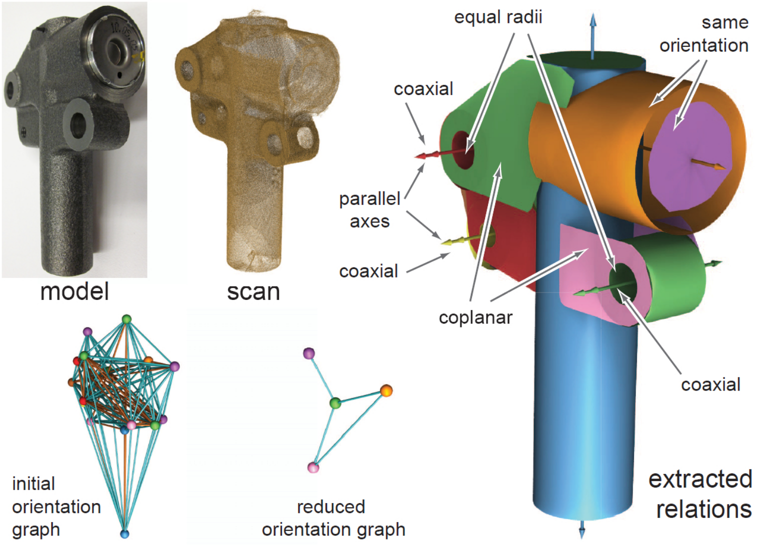


Figure 2.7: GlobFit: Global relations between shape primitives are detected to enhance the resulting 3D model. The orientation graph is used to extract a non-conflicting set of parallel and orthogonal orientations [82]. ©ACM.

world assumption can be extended to vertical structures and orthogonal intersections [22, 83] Windows and doors can be modeled with a set of shape priors [93, 124]. Furthermore, windows and doors are placed in a regular grid and their scales can be constrained to reasonable sizes [32]. Two-dimensional floor plans provide information about the localization of walls [51].

Procedural modeling techniques use shape grammars for generating architectural scenes. Usually, simple starting shapes are recursively refined, e.g. with splitting grammars. With these techniques it is possible, to generate large varieties of facades, buildings, or cities [151]. Combining procedural modeling with computer vision leads to image-based procedural modeling [93, 130, 143], where shape grammars are automatically detected in images. Procedural modeling makes it possible to easily create many variations of a reconstructed model.

Reconstructing large scenes can be simplified by first segmenting regions into semantically meaningful regions. For example, a suitable segmentation of urban scenes contains buildings, sky, ground, vegetation, and cars [78, 159]. Similarly, it is possible to segment indoor scenes into walls and furniture [55], or into predefined object classes like tables or sofas [117, 129]. Depending on the application, it is sufficient to reconstruct only the regions of interest or to use appropriate prior information for the detected regions.

A set of predefined 3D models or even large model databases can provide prior information for reconstruction. The main advantage of these approaches is that they are not limited to a few strict assumptions for specific scenes, because the set of models can be exchanged. The information from 3D models is used for extracting well-defined surfaces including sharp edges [46, 104, 161, 162] or for assembling scenes with a set of 3D models [36, 129]. In the latter case, models do not provide the exact reconstruction of a real scene, but for some applications it

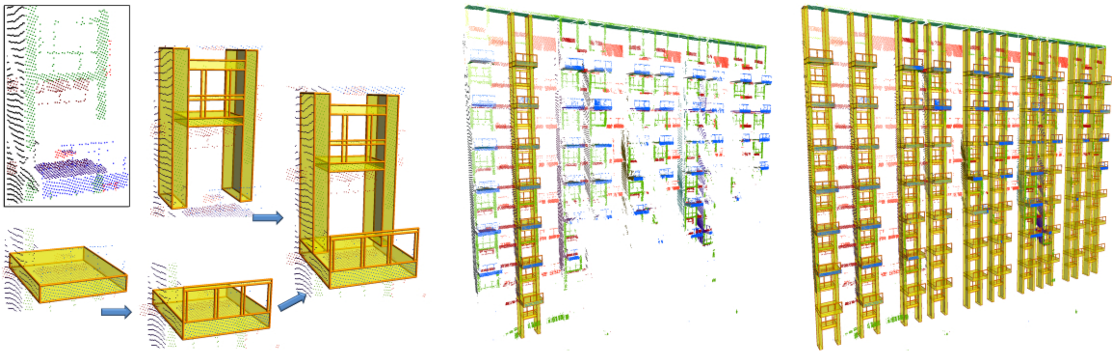


Figure 2.8: SmartBoxes: The user interactively defines models for windows and balconies. The models are duplicated to reconstruct one column, which then is duplicated to model the whole facade [97]. ©ACM.

is sufficient to have a similar scene obeying some constraints. For example, for interior scenes it is probably important that the walls are accurately reconstructed but the furniture can be represented by standard 3D models. However, the recognition and alignment of 3D models to noisy and incomplete input data is still a difficult challenge, which is not yet solved for the general case.

2.3 Point Cloud Reconstruction

In this section we describe reconstruction systems based on point sets from laser scans or range cameras. While many of these algorithms can also be applied to dense point sets from photogrammetric reconstruction, we describe algorithms specialized on photogrammetric data in Section 2.4.

2.3.1 Interactive Approaches

Chen and Chen [23] presented a semi-automatic approach for reconstructing architectural models from point clouds. The point cloud is divided into planar segments by clustering point normals and subsequently extracting planes. Boundaries are detected automatically for each plane from the assigned points. Neighboring planes are defined where points exist near the plane intersection line. The user can manually add new planes and define edges between planes. For the final reconstruction, each plane is analyzed together with its neighboring planes by generating the dual polyhedron. It is shown that corners of three or more intersecting planes are formed by a hamiltonian circuit in the dual polyhedron. The user can manually define corners for completing the 3D model.

SmartBoxes [97] is an interactive modeling tool for urban scenes in which the user loosely defines and manipulates building blocks (see Figure 2.8). The building blocks are optimized based on a data term gathered from the underlying scan data and a contextual term for respecting relations to nearby similar blocks. Smartboxes are initialized by a loosely drawn rubberband from the user and are then snapped to the underlying scan data. It is possible to group multiple boxes which then form a compound smartbox. Repetition in urban scenes is exploited by duplicating existing smartboxes to new locations. The duplicated smartboxes are automatically adjusted to the scan data while at the same time contextual relations with other boxes are utilized. The contextual relations prefer equal distances and sizes as well as collinear alignment of multiple boxes.

Arikan et al. [4] presented an interactive optimization system for fitting planar polygons to point clouds and snapping neighboring polygon elements together. An initial model is automatically generated by detecting a set of planes together with boundary polygons with a RANSAC-approach. Local adjacency relations are detected between polygon elements such as faces, edges, and vertices, which are then automatically snapped together. The user can sketch modifications of the model, which are also included into the optimization-based snapping algorithm.

Böhm [12] reconstructs facades from point sets by first fitting a plane to the facade and then generating details by projecting the point set onto a displacement map aligned with the plane. Defect areas can be interactively selected and repaired by utilizing the repetitive information of facades. Repetitions in buildings have also been used by Zeng et al. [164] for consolidating point sets. Repeating elements are selected by the user and used for denoising and filling missing data.

2.3.2 Automatic systems

Schnabel et al. [126] presented a hole filling algorithm based on multiple primitive shapes. Primitive shapes (planes, cylinders, spheres, cones, and tori) are detected in the point cloud with a RANSAC-approach [127]. The goal is to reconstruct a closed surface that adheres to the detected set of primitive shapes. The surface is computed with volumetric graph cuts by minimizing a cost function, that minimizes the surface area and penalizes areas not coinciding with primitive shapes. With this optimization, holes in the point cloud are filled by extending neighboring primitive shapes.

Jenke et al. [63] presented a reconstruction algorithm based on primitive shapes which focuses on the extraction of boundary points for each detected primitive shape. An initial boundary is extracted by analyzing the neighborhood of the points that are assigned to the primitive shape. The boundary points are then optimized such that they are close to their corresponding shape as well as to other nearby shapes. This optimization leads to boundaries located at the intersections of nearby shape primitives. If no other shapes are nearby, smooth boundaries are generated.

Another approach for primitive shapes [64] reconstructs surfaces by organizing locally smooth patches in a connectivity graph. The patches are initialized by downsampling the point cloud and connected based on their spatial distances. A patch is represented with basis functions and further divided by detecting primitive shapes when the local noise scale is high. Neighboring primitive shapes are not connected, because they are probably divided by a crease line. The patch surfaces are optimized by the following cost functions: distance between reconstructed surface and point cloud, curvatures of small patches to avoid over-fitting, and inconsistencies between adjacent patches.

Labatut et al. [77] reconstruct point sets with shape primitives such as planes, cylinders, cones, and spheres, which are detected with a modified RANSAC procedure. Due to a hierarchical detection mode, the shapes are organized in a binary-space-partioning (BSP) tree. The BSP tree splits the three-dimensional space into cells which are then divided into inside and outside cells. The BSP cells are connected in a network graph, in which edges are formed by triangulations of the shape primitives. Surface areas, which cannot be approximated by shape primitives, can be included into the network graph using a Delaunay triangulation of the points. The graph nodes are labeled as inside or outside with graph cuts. The energy function is computed from visibility constraints of each input point. The visibility information has to be provided with the point set, i.e. each point is assigned to one or more viewpoints.

Chauve et al. [22] create polygonal representations from noisy point clouds incorporating known priors from architectural scenes such as vertical structures and orthogonal intersections. Planar segments are detected in the point cloud and subsequently used for adaptively decomposing the 3D space. Nearby planes are intersected and the 3D space is divided into a cell complex. Similar to Labatut et al. [77], the cells are labeled inside or outside with a graph cut approach based on visibility constraints. The reconstruction of architectural scenes is improved by adding *ghost primitives* to the set of planes. Ghost primitives are oriented horizontally or vertically and are created at all boundary lines of detected planes.

Vanegas et al. [150] reconstruct urban structures under the Manhattan World assumption. The input points are classified into one of the three dominant surface directions and into one of four possible local shape types, namely walls, edges, corners, and corner edges. The shape type and its orientation is determined by analyzing the local neighborhood of each point. A reinforcement step improves the classification, e.g. by testing if an edge is surrounded by corresponding walls. The point classifications are used for extracting connected coplanar clusters and a volumetric approach is used for generating closed models. The inside area of the volume is extracted by raycasting it from all main directions.

Li et al. [83] use LiDAR data as input and combine it with photographs of the scene. They exploit the fact, that images have a high resolution and usually cover more areas of building facades than laser scans. The photographs are manually rectified and registered with the point cloud. Rectangular planar regions are segmented by clustering 3D points and detecting lines in the rectified input image. Depth values are assigned to all regions with a multi-labeling approach.

Wu et al. [156] presented a schematic surface reconstruction of generalized cylinders which are created by sweeping vertical profile curves along horizontal transport curves (see Figure 2.9). Transport curves approximately coincide with floor plans of a scene as they are parallel to the to the ground plane. First, the ground plane normal and transport normals for each point are extracted based on principal direction of the input points. Horizontal transport curves are generated at planes where the input points have minimal curvature. For each transport curve, multiple profile slices are extracted from the point set and merged to a profile curve at a canonical position. The curves are further optimized to generate smooth surfaces, and finally a displacement map is computed for including fine-scale geometric details.

A wide range of algorithms address the problem of reconstructing city models from airborne LiDAR data. Most of them extract foot prints of buildings and generate 2.5D models. Also hybrid approaches are available which additionally create detailed facade reconstructions from terrestrial data. Brenner [17] as well as Haala and Kada [52] provide surveys of reconstructing buildings.

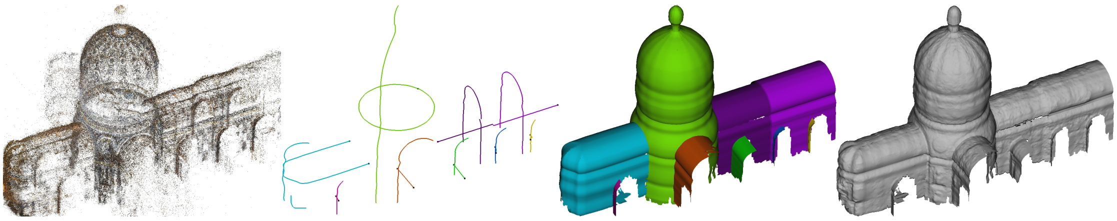


Figure 2.9: Schematic surface reconstruction (from left to right): input point cloud, transport and profile curves, swept surfaces, surfaces with displacement maps [156]. ©IEEE.

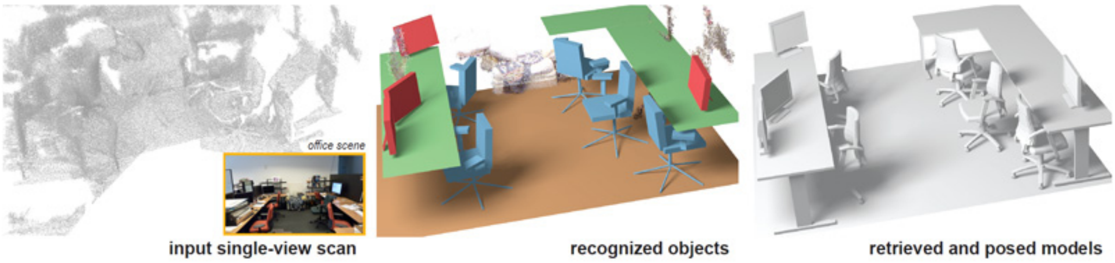


Figure 2.10: 3D models are detected in a scan and used for reconstruction [73]. ©ACM.

Lafarge and Mallet [78] reconstruct city models from airborne LiDAR data. They reconstruct standard roof sections with geometric primitives such as planes, cylinders, spheres or cones, while more irregular roof components are represented with triangle meshes. The point set is first segmented into semantically meaningful regions to separate buildings from other areas. Buildings are further segmented into geometric primitives with a region-growing approach. Finally, the shape primitives for buildings and other urban components are arranged in a dense 2.5D grid representation by propagating labels to empty cells in the grid.

In the field of robotics, it is important to provide a functional mapping for reconstructed 3D objects which is necessary for performing robot tasks. Rusu et al. [121] presented a system for reconstructing kitchens where robots need to identify handles and knobs. Point clouds are first segmented into planar region with a region growing algorithm based on normals and curvatures. Axis-aligned 2D quads and cuboids are fit to the planar regions and classified as tables, cupboards, or drawers. Handles and knobs are detected in point sets which have not been assigned to a planar region but are located close to one.

Many interior scenes contain similar objects such as tables or chairs. Recent techniques [73,98,129] therefore propose to recognize and place 3D objects from a model database in order to approximate point clouds (see Figure 2.10). The resulting models do not have exactly the same geometry as the original scene, but they contain semantic information required for scene editing and other high-level applications.

Shao et al. [129] first apply an automatic indoor segmentation algorithm on multiple RGBD-images, which can be be manually edited with a stroke-based user interface. The segmentation already assigns one of 10 semantic labels to each region, e.g. sofa or table. For each segmented region, a similar 3D model to the according depth information is located in the 3D model database. A random regression forest is used to solve this model instance recognition problem, in which 496-dimensional feature vectors, containing geometry moments, spin images, and other geometric features, are used. Training data is generated by rendering multiple depth images from different views for each 3D model.

Kim et al. [73] represent each model as a hierarchical graph structure of shape primitives, which allows to include different configurations such as open or closed drawers. In the learning phase, individual objects are scanned from multiple viewpoints and with different configurations, from which the shape primitives are automatically detected. In the recognition phase, dominant planes are extracted from a single scan and classified as ground, walls, or tabletops. The remaining points are segmented into connected components, which are matched to 3D models. Matching is done for individual parts of models based on feature height distribution, principal components, and object sizes.

Nan et al. [98] use a search-classify approach, where recognition is interleaved with a template deformation step. They exploit the observation, that man-made shapes often have a natural segmentation along their upward direction, e.g. a table can be divided into table legs and top. Objects are divided into maximally three horizontal slabs by analyzing the point distribution. The size ratios of the horizontal slabs are used together with the absolute size as feature vector. A random forest classifier is trained based on manually labeled scans. In the recognition phase, the input scan is oversegmented and a region growing algorithm is used for identifying templates.

2.4 Image-based Modeling

In this area we discuss approaches for generating 3D models from one or more input images. In case of multiple images, it is usually provided that the images are registered to each other. Additionally, for some algorithms it is necessary to have a sparse point cloud from structure-from-motion algorithms or even a dense point cloud from multi-view stereo (cf. Section 2.1.1). Especially the latter condition cannot be provided for all scenes, because it is not possible to apply multi-view stereo to texture-poor and reflective surfaces.

Automatic approaches mainly exist for outdoor scenes, because interior scenes provide additional challenges, as stated by Furukawa et al. [42]: Interior rooms consist of large texture-poor surfaces where feature-based stereo algorithms cannot compute 3D positions. It is more complicated to capture images with large overlapping areas especially in case of multiple rooms. Finally, thin structures such as doors, walls, and tables demand a high resolution relative to the scale of the scenes, which poses a scalability challenge.

2.4.1 Interactive Approaches

The Façade system by Debevec et al. [31] uses geometric primitives for image-based modeling with multiple views. The user selects a parameterized model such as a cuboid, prism, or pyramid for modeling a part of the scene. Then the user defines line segments in the images and manually links them to the corresponding edge of a model. The parameterized models are then fit to these correspondences with non-linear minimization. Poulin et al. [111] presented a modeling system,

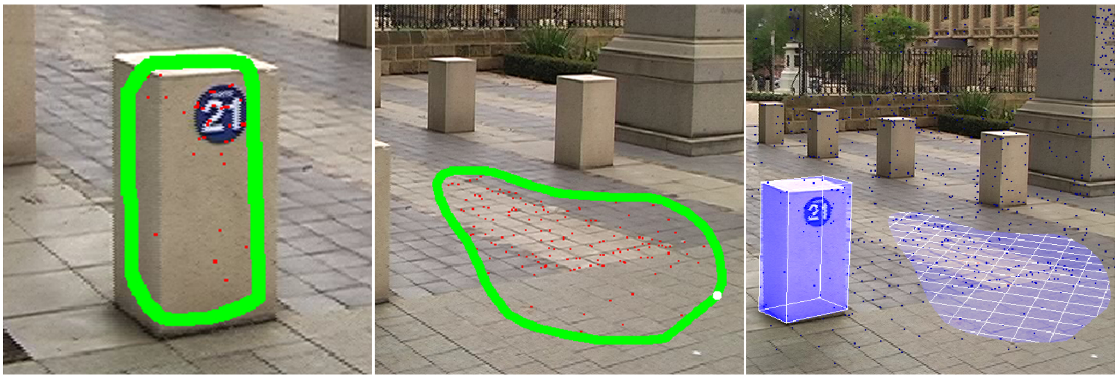


Figure 2.11: A plane and a cuboid are fit to images and the sparse point set provided by structure-from-motion [144].

where the user defines vertices of planar segments in multiple images. Additionally, it is possible to define constraints for coplanarity, parallelism and perpendicularity. Commercially available packages for generating 3D models from images [5, 141] are very similar to these approaches. The user has to define vertices or edges of objects in two or more images.

Van Den Hengel et al. [144–148] presented various techniques for interactively generating 3D models from images and videos. The algorithms have in common, that user-defined input in the images is combined with the sparse point set generated by automatic structure-from-motion. In the model-based approach [144, 145] parametric models (e.g. cubes and planes) are fit to the 3D point set and to 2D cues like intensity gradients in the images. Additionally, it is possible to define relationships between models. These relationships state e.g. that two models share a face or edge, or that multiple objects are positioned collinear. The parameters of a model are initialized by a set of 3D points selected by the user, and optimized based on a Markov Random Field.

Van Den Hengel et al. also presented the VideoTrace system [148], where the user traces the boundary of an object in video frames as a set of line segments. Each closed set of line segments is converted into a 3D object face. The user can navigate to another frame of the video and see the projected boundary of the 3D object. The user can drag line segments or corners in the new frame so that the 3D model is projected to the same contours in all frames. In the background, Videotrace estimates the 3D position and orientation of all object faces. The initial planes are estimated from the 3D points gathered by structure-from-motion. Several hypothesised models are generated and evaluated based on image information. For each object face, the difference between the color histograms in the projected areas of the video frames is used as error measurement.

Photobuilder [26, 118] is a modelling system specialized on architectural scenes. The user manually draws and matches 2D points and line segments across multiple images. The line segments are used for estimating vanishing points in the images in order to constrain the reconstructed model to the corresponding directions. The manual modeling can be supported by automatic feature matching. El-Hakim et al. [34] exploit the fact that many scenes contain

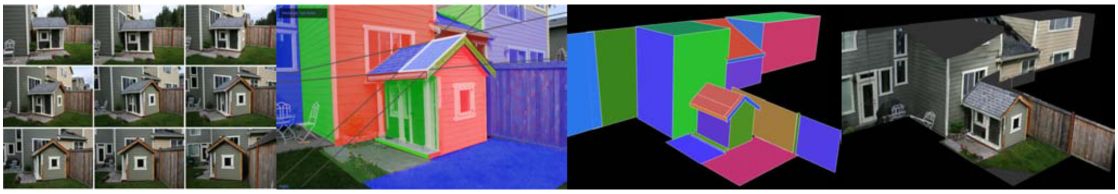


Figure 2.12: Interactive modeling from multiple images [133]. The pictures show from left to right: input images, 2D sketching interface, geometric model, and texture-mapped model.

multiple instances of the same object, e.g. windows of a building. Components are created semi-automatically from corresponding image points and can then be duplicated in other positions of the scene.

Sinha et al [133] present an interactive modeling tool for creating piecewise-planar objects from multiple images (see Figure 2.12). The user can sketch outlines of planar sections by drawing 2D polygonal outlines on images. The plane parameters are estimated from sparse points provided by structure-from-motion. An important constraint is given by vanishing points which are automatically extracted for all input images in a preprocessing step. While the user draws polygon outlines, edges are automatically snapped to directions provided by vanishing points and existing edges from other polygons. Using vanishing points makes this system suitable for architectural scenes, where models incorparate a large set of parallel lines.

Kowdle et al. [75] use an active learning approach to improve an interactive reconstruction system. The system relies on a dense point set created by multi-view stereo and is initialized by automatically detecting planes in the point set. The input images are segmented into super pixels which are assigned to one of the detected planes with a graph cut optimization. The optimization is based on the normalized cross-correlation (NCC) between corresponding super pixels of particular planes. Uncertain regions are extracted and presented to the user for support. The user connects or disconnects the regions with simple scribbles on an input image, and the optimization is repeated with the new information.

Xiao et al. [158] interactively model details of building facades from multiple views. It is assumed, that the underlying planar rectangle or cylinder of the facade is already available. The input images are projected onto the geometry and composed in a texture image. The facade is decomposed into rectilinear regions based on line segments of the composed texture. The user can adapt the segmentation with a set of operations, e.g. by adding or deleting line segments. A depth value is assigned to regions based on the 3D points from SfM. For regions without 3D points, the user can interactively assign a depth value. Depth values can be transferred to other regions and it is possible to define relative depth constraints between two regions.

Ceylan et al. [20] exploit symmetric arrangements of line segments for reconstructing details of architectural scenes. They first match line segments across the input images and fit planes to the resulting 3D line segments for generating a rough model of the scene. For each planar segment, a rectified plane texture is composed from the input images and the reconstructed 3D line segments are projected onto it. For a robust extraction of symmetries, the user sketches one element of a symmetric pattern with a few strokes. Corresponding elements are automatically detected based on normalized cross correlation (NCC) and the detected image lines. Depth offsets are computed by performing a 1D search and comparing the rendered element boundaries with the 2D edges in the input images.

Reconstructing models from single images is an ill-posed task, because no 3D information can be generated by epipolar geometry. Nevertheless, it is possible to extract spatial information from single views based on perspective cues. Criminisi et al. [29, 84] extract 3D information from single images via rectification of planar surfaces based on vanishing points. Oh et al. [103] provide a system for directly painting depth in the input image. This approach is enhanced by placing geometric primitives or special objects for organic shapes or faces.

Hedau et al. [55] recover the spatial layout of interior rooms from single images. A room is modeled by a box which is fit to perspective cues. The main challenge is that interior rooms are usually cluttered by a lot of objects. In an iterative approach, they localize clutter and refine the parameters of the box.

2.4.2 Automatic reconstruction

Werner and Zisserman [155] detect line segments and extract vanishing points in multiple images. The line segments are matched across the images with photometric constraints and trifocal geometry [125] in order to achieve 3D positions. Scene planes are detected with two methods: a plane sweeping approach along the three principal directions and a RANSAC-approach based on corresponding points and line segments. The latter approach also detects planes not aligned with the principal directions, such as roofs. Finally, protrusions and indentations are detected based on cross correlation between the images and automatically reconstructed by shape priors such as rectangular boxes and wedges. In a follow-up paper [154] more complex, parameterized models, e.g. circular arches, are fitted to the images. Selecting the right parameterized model is learned from a training set.

Dick et al. [32] reconstruct buildings with a pobabilistic model-based method using shape priors (see Figure 2.13). The models are composed of planar walls and parameterized shape primitives for other scene objects such as windows, doors, or columns. Priors on the wall layout and on the primitive parameters can be defined. These priors include e.g. scale, shape, alignment, and symmetry of primitives. Initially detected walls and shape primitives are included in an MCMC (Markov Chain Monte Carlo) sampling algorithm, which generates a range of possible solutions.

Schindler and Bauer [124] reconstruct buildings from images by first creating a coarse piecewise-planar model, and then fitting pre-defined shapes for windows and doors. A dense point cloud from multi-view stereo is used for detecting the main planes of the scene with a RANSAC-approach. Rectangular regions, that do not belong to the planes, are detected with an axis-aligned line sweeping, where the distances of the 3D points to the plane are evaluated. For each of these regions, pre-defined shapes for windows and doors are selected and fit to the input data. The shape fitting uses the segmented 3D points as well as lines and ellipses detected in the input images based on a modified Canny edge detector.

Furukawa et al. [42] reconstruct interior scenes automatically by invoking the Manhattan world assumption. As input, they use a set of images together with depth maps computed with Manhattan world stereo [41], which is described in Section 2.1.1. The depth maps contain

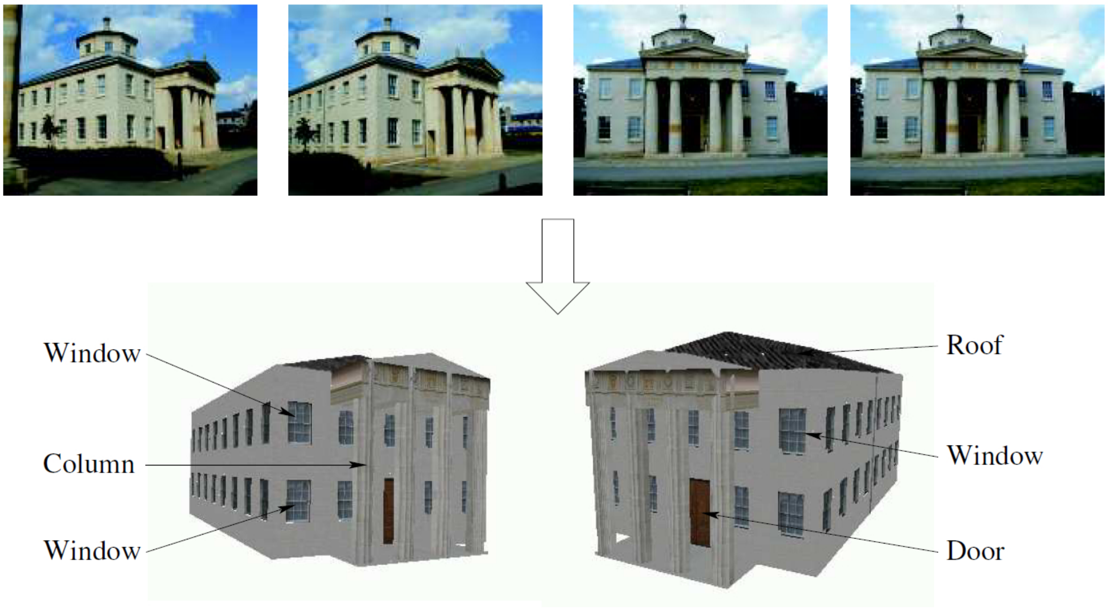


Figure 2.13: Buildings are reconstructed with parameterized shape priors [32]. ©Springer.

surfaces aligned along three dominant axes of the scene. The depth maps are integrated by dividing a voxel grid into interior and exterior space with a graph cut solution. A minimum volume constraint is applied for generating clean edges and corners.

Bartoli [10] detects planes from the sparse point reconstruction with a RANSAC approach. Then a modified bundle adjustment is used for simultaneously optimizing the plane parameters, 3D point positions, and the camera parameters. The optimization uses a photometric criterion, where the original images are compared to the predicted projections from the current model parameters.

Fraundorfer et al. [39] reconstruct piecewise-planar scenes by establishing homographies between pairs of images. Initial homographies are estimated by matching image regions [66, 90]. The regions are enlarged by extrapolating the homographies to neighboring pixels. New correspondences are detected and used for updating the homography. This approach is repeated until the homography parameters have converged.

Procedural modeling is suitable for building facades, because they usually contain regular structures. Müller et al. [93] automatically subdivide a single, rectified facade image into floors and tiles. Facade tiles are further subdivided in order to extract windows which are then reconstructed with appropriate 3D elements from a model database. Depth values of the facade elements are defined by the user. Finally, the computed subdivision are encoded as shape grammars.

Simon et al. [130] automatically reconstruct procedural modeling rules from multiple images and a point cloud. An evolutionary algorithm is used to detect and optimize shape grammars for facades as well as for 3D building models under two energy models. First, the derivation should induce an optimal semantic partition in the input images which is learned from training data

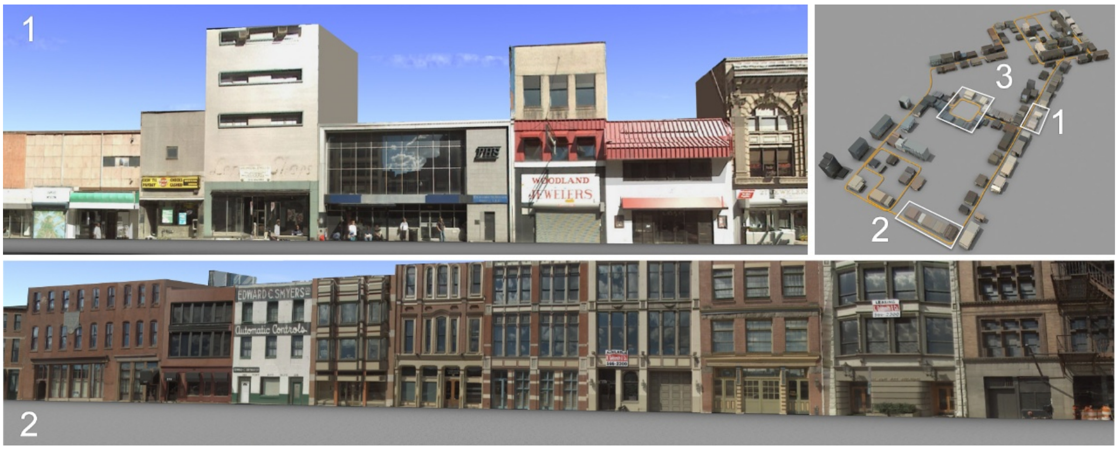


Figure 2.14: Reconstructing building blocks from street-side imagery [159]. ©ACM.

created by a user. Second, the resulting 3D model should coincide with depth maps generated from the 3D point cloud.

The system presented by Xiao et al. [159] is specialized on facades and street-side imagery (see Figure 2.14). They first classify the building regions in the input images for segmenting them from other areas such as vegetation or cars. The classification is optimized over all input images in order to have a consistent segmentation. Lines are extracted and matched over the input images. Vertical lines are used to divide the scene into multiple building blocks and horizontal lines are used to align the facades to a local coordinate frame. For each building block, a color image and a depth map are generated from the input images and a dense point set. The depth and color images are used for analyzing the facade for structural elements at different depths. Rectangular elements and repetitive structures are detected automatically and modelled with the according values of the depth map.

Xiao and Furukawa [160] combine multiple photographs with laser scans to reconstruct large indoor environments. They propose a new algorithm called *Inverse CSG* where they automatically create a constructive solid geometry (CSG) representation consisting of cuboids with union and difference operations. The 3D space is first split into horizontal slices which have the same horizontal structure and usually denote one floor of the building. A 2D CSG model with rectangles is created for each slice. The CSG model has to be consistent with the free space and the captured points, and it should have a small perimeter. Finally, 3D primitives are generated from the 2D reconstructions and a 3D CSG model is created.

CHAPTER 3

Image-Based Modeling of Industrial Environments

In this chapter, we propose an interactive technique for reconstructing buildings solely from images. The algorithm is aimed for scenes with textureless and specular surfaces, where automatic methods fail. It is sufficient to approximately define the position of outlines and edges of simple geometric objects, which are then automatically placed in the scene.

3.1 Introduction

Multi-view stereo algorithms automatically reconstruct a scene from a set of images [43]. These methods usually create three-dimensional objects with a lot of polygons and depend on short camera baselines as well as diffuse surfaces. On the other hand, there are photogrammetric applications [5, 141] for manually creating 3D objects based on photogrammetric methods. They need a lot of user input, mainly creating point correspondences across multiple views, but in exchange they are independent from lighting conditions and camera baselines. Another advantage is that the resulting 3D models are very simple and only have a few polygons. A drawback is that calculations are very error-prone if the user makes a mistake or the user input is not accurate enough.

We present a new modeling technique which keeps the advantages of manual photogrammetric modeling but makes the workflow more convenient for the user. We combine the expertise of the user with automatic operations in photogrammetry and image processing. This will take workload off the user and increase the stability and accuracy of the reconstruction system. Figure 3.1 shows an input image and the result of our modeling system for a set of oriented images.

The proposed modeling technique is especially useful for scenes with specular and textureless surfaces, such as industrial buildings. For these scenes, it is not possible to automatically extract reliable point correspondences across images. For this reason, there are not enough 3D positions to initialize and optimize 3D objects based on 3D sparse reconstruction [133, 144].

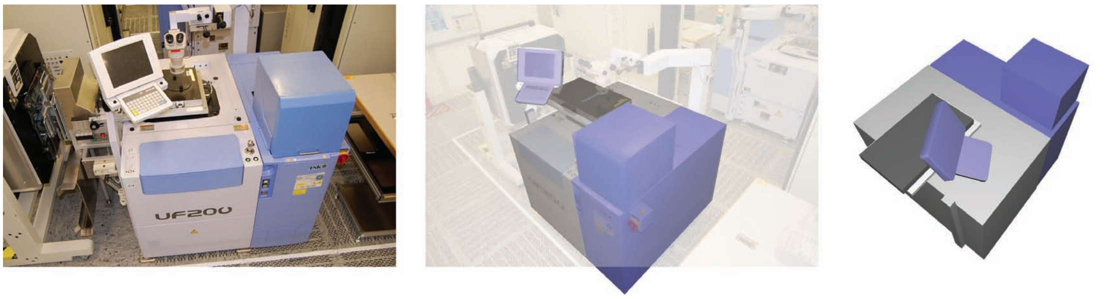


Figure 3.1: The images show (from left to right) an input image and the resulting model overlaid to another input image and from a new perspective.

The application of multi-view reconstruction methods often depends on the input images. Therefore, we provide a set of different methods which can be used together by the user. If automatic approaches fail, it is important to have a fallback alternative, even if it requires more input from the user. The goal of our application is to make the workflow fast if it is possible, but it has to be successful in any event.

3.1.1 Problem statement

The primary goal of our multi-view reconstruction system is to get information about the size and general shape of industrial machines. This information is used for calculating the required space for machines as well as for visualizing rebuilding plans. Small details of the facilities are often not important and sometimes even disturbing for planning the arrangement of machines. Deciding which objects are important strongly depends on the reconstruction goals and therefore should be left to the user.

Individual elements of artificial objects are often quite simple and can be approximated with geometric primitives like cubes or cylinders. These objects should be reconstructed as simple as possible because dense vertex geometries are often not desired in the CAD-workflow. For small elements, e.g. small switches, it is often sufficient to be included only with texturing in order to obtain the simplicity of the models. On the other hand, if they are important, the user should be able to explicitly reconstruct these small objects.

Another common type of objects in industrial environments are tubes and pipes. It is hard to automatically match features of tubes across multiple views because they often have very homogeneous surfaces. Just like other objects, the geometry of tubes should be kept as simple as possible.

An important prerequisite for 3D reconstruction is to have accurate camera poses for all input images. Structure-from-motion algorithms require correspondences of points or lines across multiple images. Industrial sites often contain objects with highly reflective surfaces or poorly textured objects. Bad lighting conditions or taking photos with a flash should not affect the camera orientation. This makes automatic feature detection and matching very difficult because the scene does not provide enough view-invariant surface patches. Automatic matching is further complicated if only a few images should be sufficient for the reconstruction and therefore the camera baselines are very wide.

The following list summarizes the requirements of our image-based modeling system:

- simple workflow on-site
- wide camera baselines
- objects may contain reflective surfaces
- should work with textureless objects
- independent from lighting conditions
- fast and accurate camera orientation
- models with simple geometry
- intuitive user interface

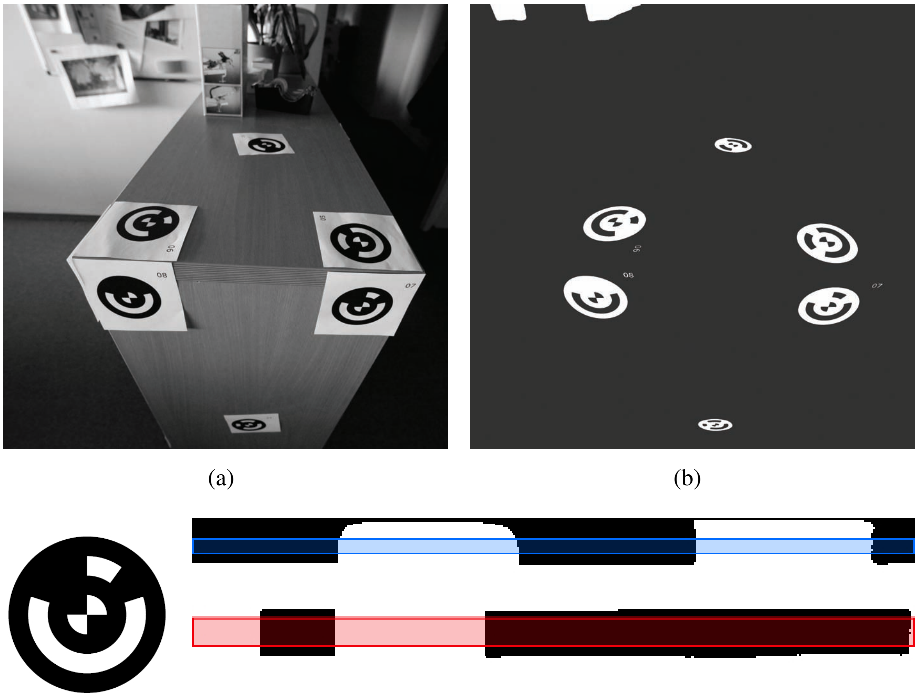
The workflow of the reconstruction system can be divided into camera orientation, image segmentation and image-based modeling. Camera orientation uses coded markers in order to achieve short computation time and accurate results. The next step is to segment regions and edges of objects or object parts in multiple images with interactive image segmentation techniques. On the basis of oriented cameras and image segmentations it is possible to fit parametric models to the scene. The user selects an appropriate model type, multiple image segmentations and optionally additional constraints and the model parameters are automatically fit to these requirements. Complex objects can be modelled by fitting primitives to individual parts of the object.

3.2 Camera Orientation

An accurate camera orientation is important for successive reconstruction steps because errors and inaccuracies will propagate through the reconstruction pipeline. Additionally, the calculation of extrinsic camera parameters has to be fast in order to get the orientation results on-site. If additional photographs are needed they should be taken as soon as possible. Long time intervals between taking photographs increase the possibility of undesired changes in the scene.

We use pre-calibrated cameras in order to increase the accuracy and stability of the camera orientation calculations. Coded markers are used for localizing a planar calibration pattern in the images. The correspondences between 2D and 3D positions are used for determining the intrinsic camera parameters as well as the radial and tangential distortion parameters [163].

We use point-to-point correspondences for calculating the extrinsic camera parameters which are position and orientation. Coded markers in the scene make the matching of points across multiple views very accurate and simple (see Figure 3.2). Points can be manually marked in the images if coded markers cannot be placed in the scene. If the objects in the scene are textured and they have diffuse surfaces, it is possible to automatically match natural features, e.g. with SIFT points [88]. Of course, coded markers, manual points and automatic feature points can be used in combination. In this case the contribution of coded markers is weighted higher because they are less error-prone than other points.



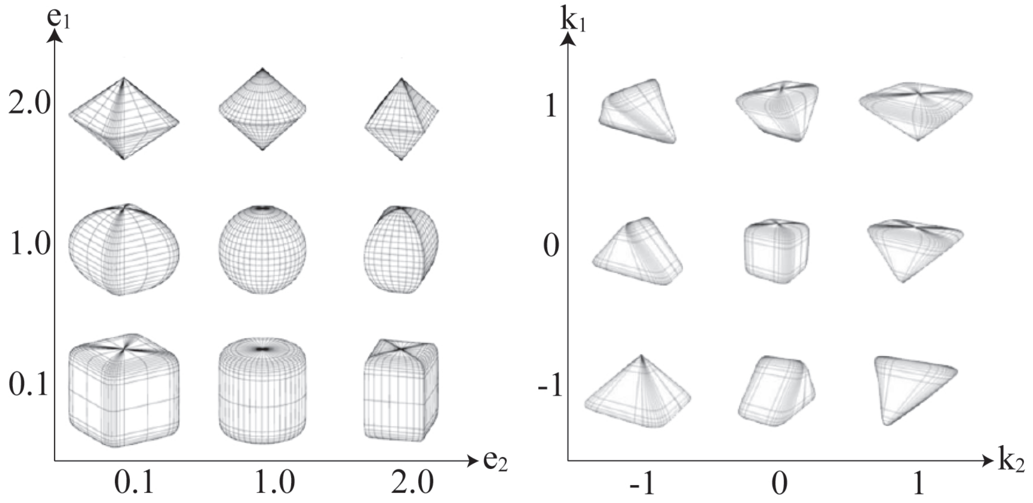
(a) (b)

(c) (d)

Figure 3.2: Detecting coded markers in an image. (a) Input image for marker detection. (b) Detected MSERs. Only regions with the shape of an ellipse will be used. (c) Original marker which is placed in the scene. (d) Unwrapped marker: The top section contains the x-corner in the center. The bottom section contains the visual bit code of the marker.

The relative pose is computed for a pair of images with the five-point-algorithm [101]. All other images are sequentially added to the network with the three-point algorithm for correspondences between 2D points and known 3D points. The resulting camera network is finally optimized with bundle adjustment [53]. The triangulated positions of point correspondences and the camera parameters are optimized with a sparse Levenberg-Marquardt algorithm which minimizes the distances between reprojected triangulated positions and the according image positions.

The accuracy of pose estimation can be enhanced by using measurements from a total station, which measures 3D positions with a laser signal. The survey points provided by the total station are matched to image points, either manually or the survey points have already been attached to coded markers during acquisition. The absolute pose of cameras can be calculated directly with these 2D-3D point correspondences without using the relative orientation between two starting cameras.



(a)

(b)

Figure 3.3: (a) Basic shapes of superellipsoids with uniform scaling. (b) The effect of tapering on a cube-shaped super-ellipsoid.

3.3 Model Fitting

Our image-based modeling technique fits parametric models to different constraints that come from image segmentation or other user input. The user creates an object from a specific model type and adds image segmentations and other constraints for optimizing this model. Complex objects are reconstructed by combining multiple primitives. The connectivity between these primitives can be ensured by applying constraints to multiple models.

We first describe the possible models and the constraints. Then we describe how these elements are combined during model fitting. Finally, we present special considerations for the reconstruction of tubes.

3.3.1 Models

We provide models for the geometric primitives cube, box, sphere, cylinder, pyramid and frustum. Additional shapes are provided by superellipsoids. More complex objects can often be created by combining multiple primitives. A special case are tubes which are represented by the control points of a 3D spline.

All models are defined by a set of parameters which are modified during the model fitting process. The parameters include the pose of the model in world space and the actual shape of the model. The pose in world space is defined by position, scale and rotation, each represented by three parameters. The scale and rotation of some models can be defined with less parameters. Table 3.1 summarizes the number of parameters for each model type.

Superellipsoids [62] can be used for modeling a wide range of shapes including spheres, cubes, cylinders as well as shapes in between and non-uniform scaled versions. Superellipsoids contain two parameters e_1, e_2 in addition to the common parameters position, scale and orientation. A global deformation transformation called tapering [9] is used to represent even more shapes including pyramids, cones, frustums of pyramids and cones, wedges and all shapes in between. Tapering needs two additional parameters k_1 and k_2 which reduce the size of the shape along two directions. Possible shapes formed by superellipsoids can be seen in Figure 3.3.

Tubes are represented by a 3D spline and a diameter defining its thickness. A 3D curve is interpolated through a set of control points. These control points can be altered during the model fitting process. The number of control points is automatically defined depending on the length of the reprojected curves in the images. More details about the reconstruction of tubes can be found in Section 3.3.5.

Name	p	s	r	
Cube	3	1	3	0
Box	3	3	3	0
Sphere	3	1	0	0
Cylinder	3	2	2	0
Pyramid	3	2	3	0
Frustum	3	2	3	4 (smaller base position/scale)
Superellipsoids	3	3	3	4 (coefficients and tapering)
Tube	3	1	0	n (control points)

Table 3.1: Number of parameters per model type: position p, scale s, rotation r, and additional parameters

3.3.2 Image Constraints

The image projections of a model provide several constraints in order to find the correct parameters of the model. These constraints are created by points, edges and regions from multiple images. Edges and regions are defined by interactive image segmentation, while points can be selected from the point set already used for camera orientation.

We provide a set of interactive segmentation techniques for defining edges and regions. In our application we use interactive graph cuts [15] for region-based segmentation and intelligent scissors [92] for defining edges and contours. It is also possible to directly draw edges and region outlines in the images. The selection of an appropriate segmentation technique depends on the expected properties of the input images. Automatic segmentation techniques can also be used, provided that objects can be easily distinguished from the background and from each other.

The goal of segmentation is to define outlines and edges of the desired 3D objects. It is not necessary to define all possible segmentations in one image, but rather to provide enough constraints across multiple images for fitting a model. Not all types of segmentation have to be provided, especially because outlines are often hard to segment due to occlusions from other objects. If the user is not satisfied with a fitted model, it is possible to add more constraints and further optimize the model.

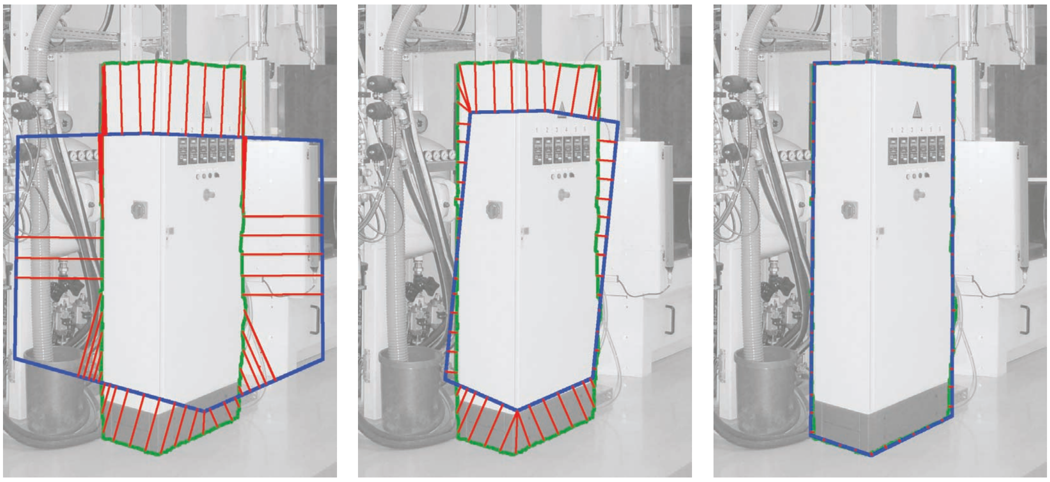

(a) (b) (c)

Figure 3.4: Optimization of an outline constraint. The distances between samples (red) of the user-defined outline (green) and the model outline (blue) are minimized during optimization.

Outlines Outlines are a very useful constraint for defining the position and scale of an object. Sample points are taken from the user-defined contour at equal distances. Using all pixels of the contour would increase computation time, whereas the result is not improved because nearby points provide almost the same information for the model fitting. On the other hand, using not enough samples increases the influence of outliers in the user-defined region.

The outline of a 3D model is projected onto the image, either by an analytical computation or by retrieving the contour of the projected model on the image. The distances between sample points and the nearest points on the model projection contour are minimized with

$$\min \sum_{I} \sum_{p} d_2(p, outline(I))^2, \tag{3.1}$$

where d_2 is the Euclidean 2D image-based distance, p denotes a sample point on the user-defined contour and $outline$ calculates the outline of a model for a specified image I. Figure 3.4 shows the distances at various minimization iterations.

Edges Edges either correspond to the projections of model edges or to the outline of a model in an image. The user can segment partial edges, and also multiple edges at once. Minimization is done in a similar way as for outlines. All visible edges of the model as well as the outline are calculated for the camera of the current image. The distances between sample points of the user-defined edges and the nearest projected object edge are minimized with

$$\min \sum_{I} \sum_{p} d_2(p, visEdge(I) \cup outline(I))^2. \tag{3.2}$$

Points Triangulated points from point correspondences can be selected as constraint for the corners of a model. In case of using manual points for computing the camera poses, corners are often used because they can be seen and identified easily across multiple views. The 3D positions of the 2D points have already been calculated and optimized by triangulation and bundle adjustment and form a very accurate constraint for the model.

Points define a constraint in three-dimensional space in contrast to edges and regions which define constraints in image space. For each 3D point the nearest corner of the model is detected. The model is fitted by minimizing the distance between the points and the model corners with

$$\min \sum_{P} d_3(P, P_M)^2,$$

where d_3 denotes the Euclidean distance in 3D space, P is a triangulated point from image point correspondences and P_M is a corner of the model. Fitting to the nearest model corners can result in an undesired local minimum. Section 3.3.4 describes the initialization of the models in order to avoid such local minima.

3.3.3 Relational Constraints

The user can select relational constraints in addition to the image segmentations. These constraints form relations between multiple models or they demand certain shape properties of a model. A very simple constraint is to set the parameters of different models to the same value. For example, if two identical objects appear in a scene, they can be modeled by applying this constraint to all parameters except position and orientation.

Up Vector This constraint ensures that objects are oriented in the same direction, but the rotation around this direction can vary. We do not provide a global up-direction in the reconstruction system, because the cameras may be registered to an arbitrary coordinate system. Hence, the up-vector is only defined by the models assigned to this constraint and do not need to point to a certain direction. The constraint minimizes the angle between the up-vectors of all models with each other.

Some models can be parameterized with different direction vectors and keep the same shape. For example, the orientation of a box can be interchanged in all main axes as long as the according scale values are adapted. We reparameterize all models that they approximately have the same up-direction. Then we apply a constraint on the model rotations by fixing the according rotation parameter to the same value.

Ground Plane Objects are often located at the same ground plane. The ground plane is computed for all models assigned to the constraint by taking the lowest point of each model along its up-vector. The ground plane is fitted to these points and updated at every iteration during minimization. The constraint is then satisfied by minimizing the perpendicular distance between the plane and the lowest point of each model along the plane normal. Thus, the position of the models is adapted such that they share a common ground plane.

3.3.4 Initialization and Optimization

We use Levenberg-Marquardt [87] for the nonlinear optimization of several constraints. It is important to find a good initial model as Levenberg-Marquardt only converges to a local minimum. The position of a model can be found quite easily by triangulating the centers of segmentations in every image. For estimating the scale of the object we compute the median distance between the center and the outer contour of all segmentations in an image. This distance is projected back to the model position and the average value from all images is taken as uniform scale value. This is obviously a very rough initialization, especially for elongated objects, but the scale values usually converge very fast. Initializing a model with anisotropic scale values would increase the effort for finding the right model orientation.

The orientation of the object as well as additional parameters are more difficult to define in advance. Therefore a set of models is initialized with random parameters. They are optimized for a few iterations and the model with the lowest residuals is selected for further optimization.

We first optimize the individual cost functions from image segmentations with Levenberg-Marquardt which are independent for each model. Then the models are reparameterized according to the assigned relational constraints (e.g. to have a common up-vector). The last step is to fit all models simultaneously to all constraints.

The previous sections described the individual cost functions, which either come from image segmentations or from user-defined constraints. The goal of optimization is to minimize the sum of all individual costs. All constraints serve as soft constraints, that means solutions slightly off the constraint are also accepted. Weights can be assigned to constraints in order to increase their importance compared to others.

Models with many parameters, e.g. superellipsoids, usually require more constraints, i.e. more image segmentations, than models with only a few parameters. Models which depend highly on a specific rotation often need more initial guesses and therefore longer computation time.

3.3.5 Reconstruction of Tubes

Tubes are represented by the control points of a 3D spline and a diameter for its width. The user input and the initialization of the models are different to other models. Once the initial parameters are available for the tube, optimization is done as for all other models.

There are three possibilities for defining image constraints for a tube. All methods lead to a 2D curve in the center of the tube and a value defining the width of the tube at a certain 2D position (see Figure 3.5).

- One user-drawn curve in the center of the tube and one line specifying the width of the tube.
- A region which corresponds to the outline of the tube. It is also possible to segment only a part of the tube, if it is partially occluded. The skeleton of the region is calculated and smoothed by sequential thinning with structuring elements [138, p. 675]. The width of the tube is determined by the distance between the skeleton and the region border.

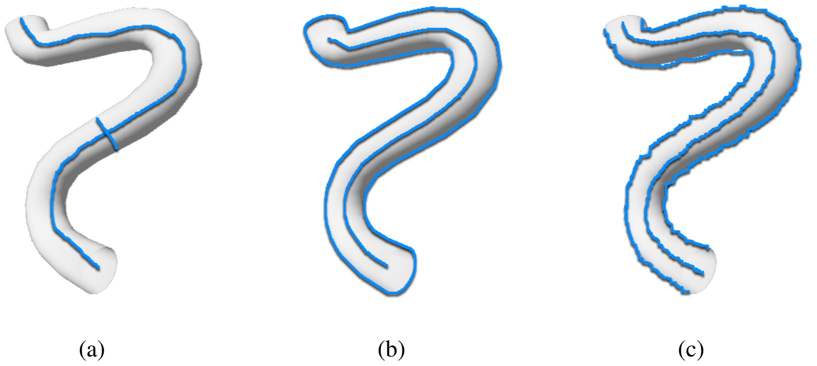

(a) (b) (c)

Figure 3.5: Segmentation of tubes: (a) user-drawn curve, (b) region with skeleton, (c) two edges and intermediate curve

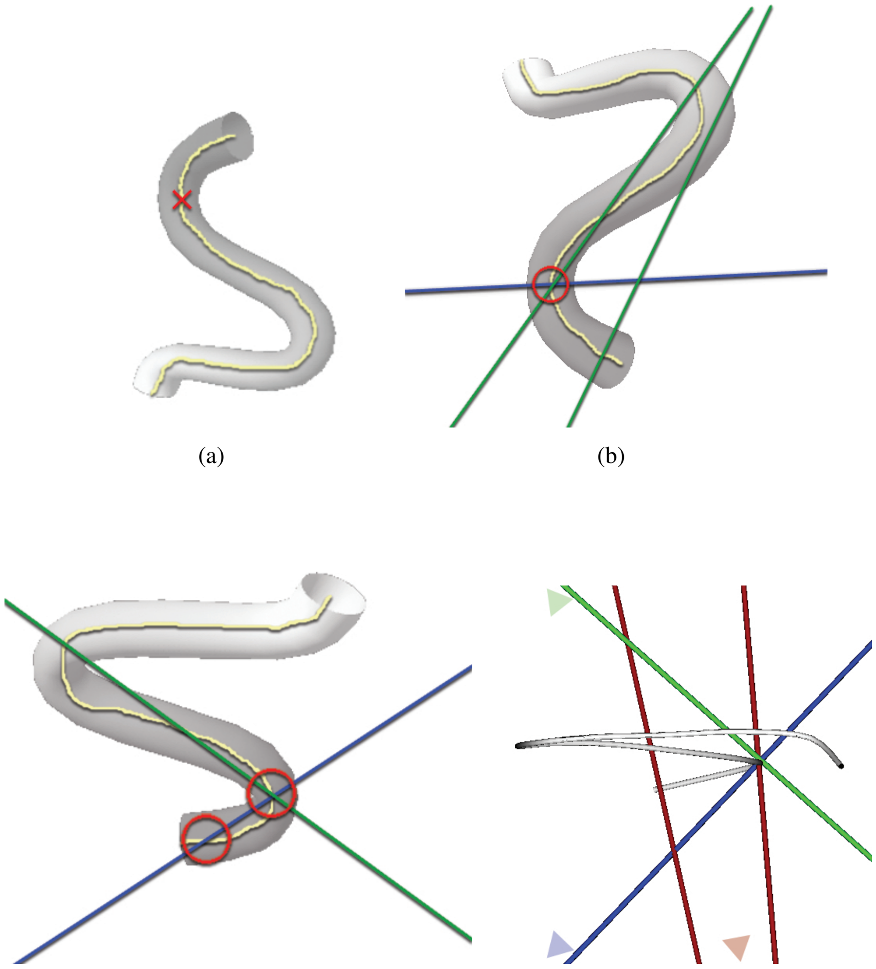
- Two edges along the outlines of the tube. The edges do not have to denote the whole tube, but the two edges should approximately mark the same segments of the tube. The central curve is approximated in the center of the input edges.

Initial 3D positions on the tube are created by triangulating point correspondences across multiple views. Tubes are often made of monochromatic and specular materials and thus do not provide any image features which could be used for matching points. Therefore, initial matches are created solely based on the input 2D curve and epipolar geometry.

Epipolar geometry states that for a point in one image, the corresponding point in another image has to lie on the epipolar line, which is the projection of the 3D ray between the point and the camera center. In our case, a corresponding point in another image has to lie on the crossing of the 3D-projection of the tube and the epipolar line. If the epipolar line meets the 2D curve in another image only once, the probability for a correct point match is very high. Of course, this is only true if the whole tube is visible and has been segmented by the user.

If the epipolar line crosses the 2D curve in more than one point, it is not possible to determine the correct match if there are only two images. If there are more images, all possibly corresponding points in all other images are computed. The corresponding points are validated by computing their epipolar line for every other image. Corresponding points are only accepted, if the epipolar lines from points from different images meet in the same point. This procedure is visualized in Figure 3.6.

It can happen that points are incorrectly matched, especially if the tube is only segmented in two images. Points which are neighbored in one image, are probably also neighbored in another image. Some points may be missing in another image, but in any case, a sequence of points must appear in the same order in all image. Points which occur in different sequences in different images are removed. The last step is to fill in additional points into large gaps along the 2D curves. The neighboring points provide clues for deciding between possible corresponding points. Finally the control points are re-parameterized at equal distances in 3D space. Figure 3.8 shows two examples for reconstructing a tube.



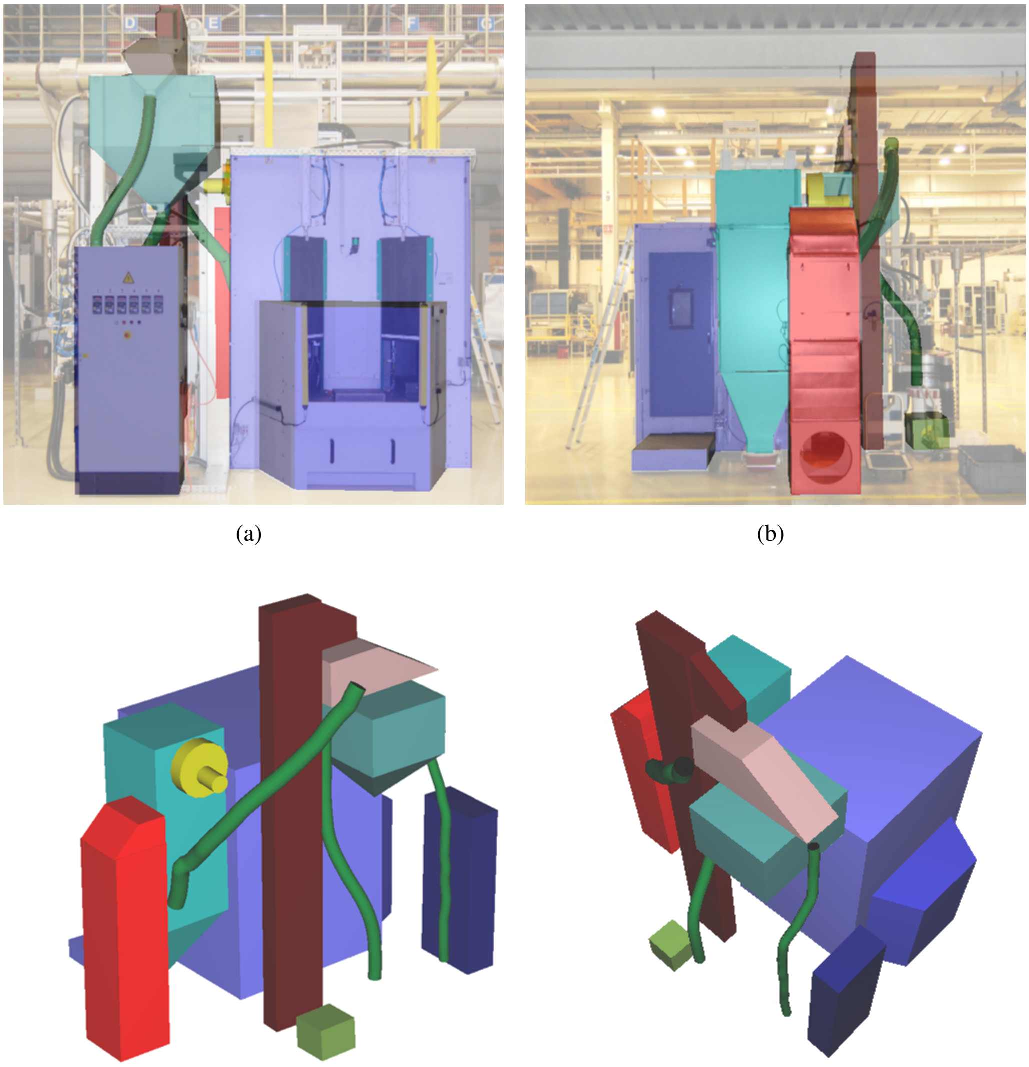
(a) (b)

(c) (d)

Figure 3.6: Initialization of tubes: (a) Sample point in image 1. (b) Image 2: The blue epipolar line from image 1 led to one corresponding point. This point is confirmed by a green epipolar line from image 3. (c) Image 3: The blue epipolar line from the first image led to two corresponding points, but the green epipolar line from image 2 validated only one point. (d) Reconstructed tube and epipolar lines from another viewpoint. The blue line originates from image 1, the green from image 2 and the red lines from image 3. It can be clearly seen that the intersection of the blue and red line is not located on the 3D curve.

3.4 Results

Figure 3.8 shows image segmentations and results for single models. An example where multiple models were used for reconstructing a scene can be seen in Figure 3.7. The scene was modeled with 19 primitives based on 9 images. The optimization contained 376 parameters and 21075 measurements from image segmentations and other constraints. The optimization of



(a) (b)

(c) (d)

Figure 3.7: This scene was reconstructed with 19 models based on segmentations in 9 images.

all models, i.e. the initialization and individual optimization of each model and the combined optimization of all models, was done in 9 minutes. The proposed algorithm is not limited to industrial environments, but can also be applied to buildings and interior scenes (see Figure 3.9).

3.5 Conclusions

We presented a new reconstruction system for modeling industrial facilities from a set of images. The system solves a specific area of reconstruction applications. Due to the textureless

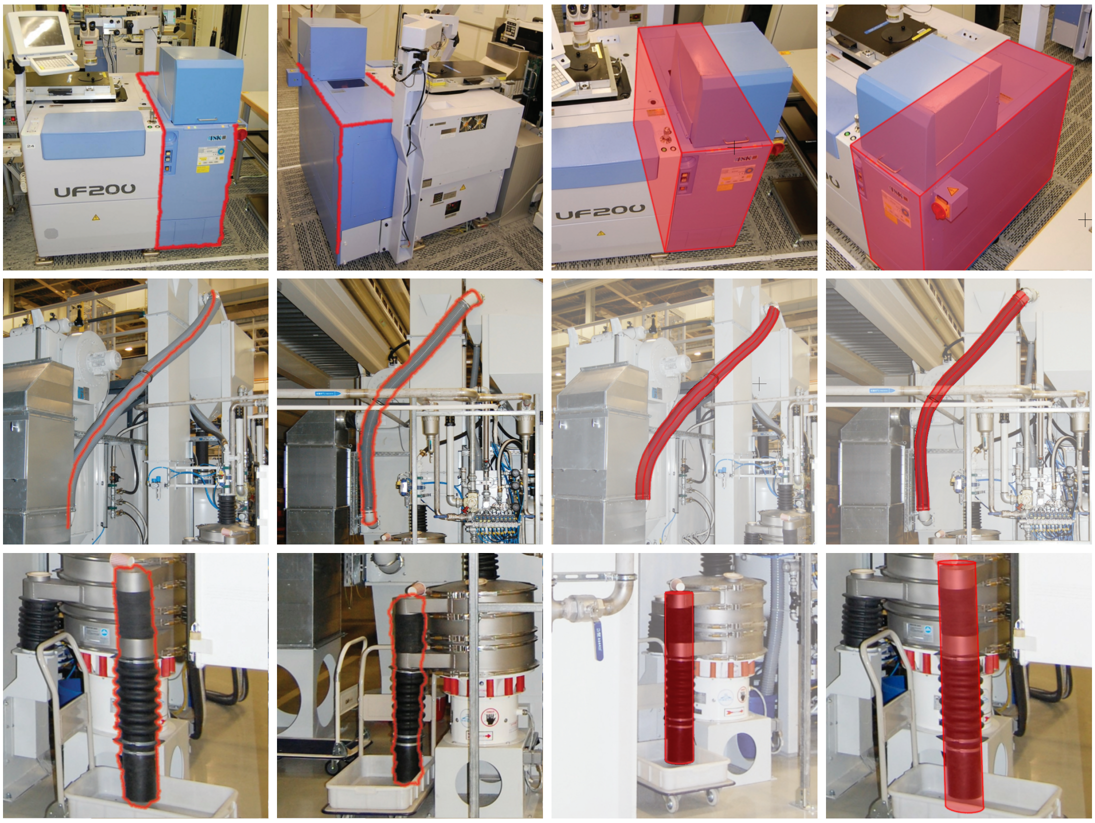


Figure 3.8: Top row: Edges in two images are used for modeling a box. The images provided 219 resp. 165 sample points. Middle row: A tube with 30 control points is modeled by an edge (181 sample points) and a region (277 sample points) in two images. Bottom row: Outlines in two images with 106 and 126 sample points are used for modeling a cylinder.

and specular materials that are typical for industrial scenes, it is not possible to rely on automatic matching techniques. For this reason, we decided to use an interactive approach, where image segmentations are provided by the user. However, the user is supported by automatic computations and the reconstruction system is easy to use and does not require any knowledge about 3D modeling or photogrammetry.

We have shown, that the restriction to parametric models is useful for industrial environments, but also for other scenes such as buildings or interiors. We have used simple geometric primitives such as cubes or spheres, as well as superellipsoids which provide a very large range of different shape. In the special field of industrial environments, tubes are an important object, for which we provided specialized algorithms for segmentation and initialization. Model parameters are automatically optimized according to user-defined constraints. These constraints are either based on image segmentations or they define relations between multiple models.

For other scenes than industrial environments, the main drawback of our reconstruction sys-

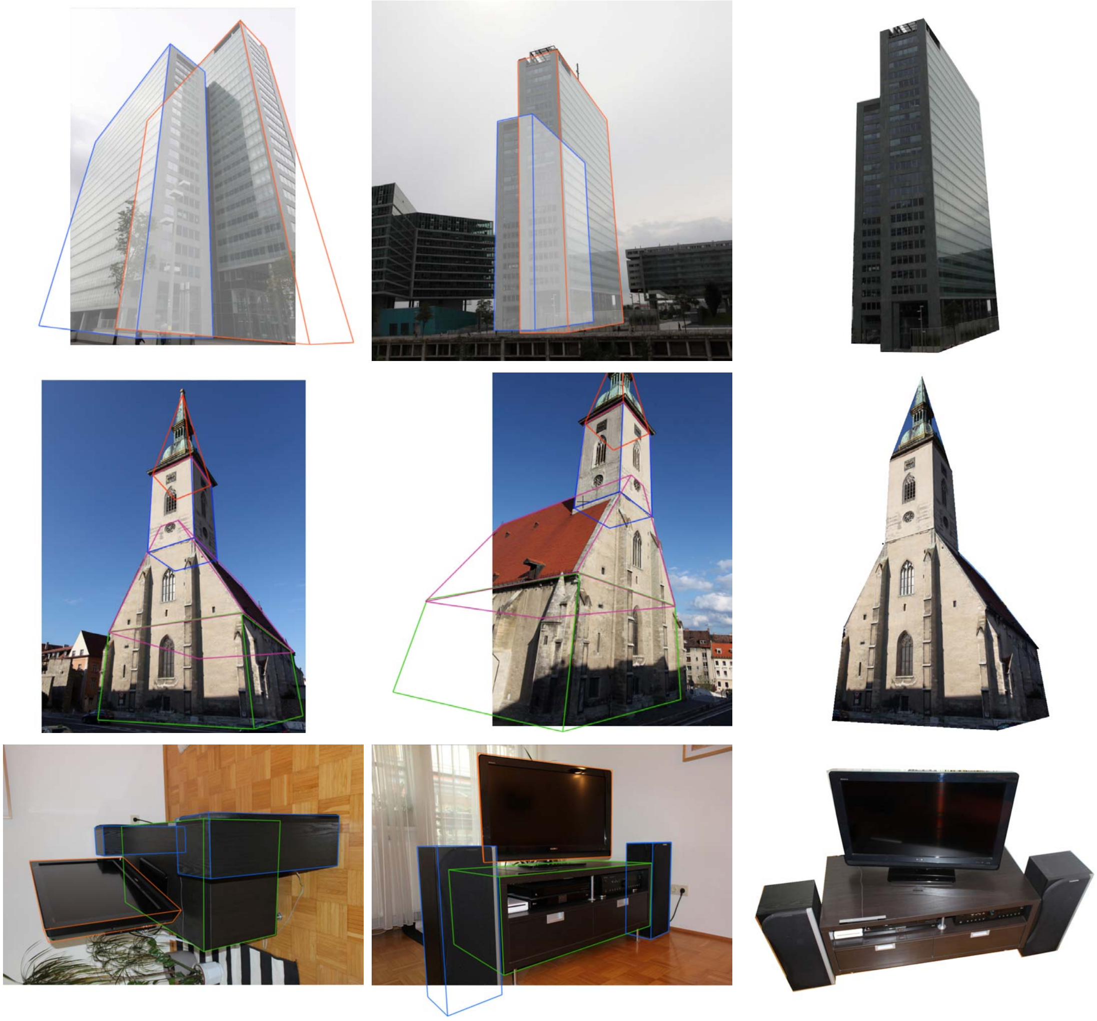


Figure 3.9: Top row: Edges in two images were used for generating the building model containing two boxes. Middle row: The church has been reconstructed with boxes, a frustum and a pyramid. The shape of the roof is too complicated for our model fitting approach, but it is possible to provide a simple approximation. Bottom row: The TV-set has been reconstructed from three images with three boxes and one frustum.

tem is the need for user-defined segmentations of the input images. While this is necessary in case of difficult materials and lighting conditions, it could be replaced by automatic detection algorithms when extensive 3D information is available. Nevertheless, the presented algorithm works well for a large set of input scenes and can be used as fallback solution for failure cases of automatic algorithms.

CHAPTER 4

Automatic Building Reconstruction from Point Clouds

In this chapter, we focus on the reconstruction of buildings and therefore exploit common characteristics of architectural scenes. Many buildings can be approximated with piecewise-planar models and the planes are orthogonally aligned. In contrast to the previous chapter, extensive 3D information is available in the form of an unstructured point cloud. Therefore, it is possible to automatically detect scene planes and create 3D models without any user input.

We introduce a novel method for automatically reconstructing low-polygonal meshes from point clouds. The point cloud is analyzed for extracting globally consistent surface normals, which are then used to robustly detect planes. If oriented images of the scene are available, it is possible to extract strong image edges which can be used as boundaries for the planes.

4.1 Introduction

Recent advances in scanning technologies allow the acquisition of point clouds from a large variety of scenes. Laser scanners are able to capture point clouds of real-world scenes, ranging from small objects to whole cities. Photogrammetric tools generate dense point clouds from a set of images [45], which works well for outdoor scenes. Although capturing methods have different advantages and drawbacks, the generated point clouds usually share the same problems of noisy and missing data, e.g. due to a limited area of capturing viewpoints. These artifacts make it very difficult to apply direct surface reconstruction methods [3, 69], where the point cloud is approximated by a large number of triangles.

Our system overcomes these difficulties by reconstructing piecewise planar surfaces from both a point set and multiple images. While point sets provide extensive 3D information, images contain more accurate edges and boundaries. By combining these advantages, our approach is suitable for quickly generating textured low-poly 3D models of buildings in a typical urban scene.

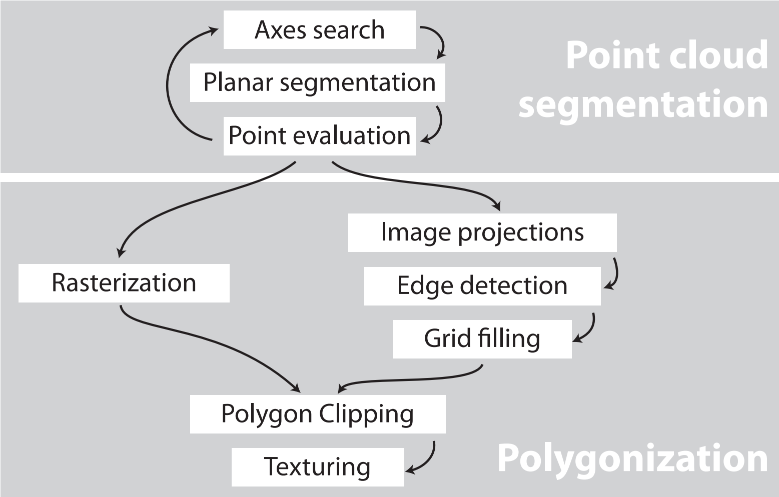


Figure 4.1: Pipeline of our proposed method: There are two different algorithms depending on whether images of the scene are available.

A remarkable aspect of our reconstruction system is that it can be used fully automatically for a wide range of architectural scenes. At the same time, it is very easy to manually intervene during the process for improved results. This is traced back to the fact that individual steps are calculated fast, intermediate results can be individually re-calculated, and customizable parameters are easy to understand.

As depicted in Figure 4.1, our system starts with segmenting the input points into planar clusters. A polygonal surface of each cluster is subsequently reconstructed by a grid-based evaluation exploiting multi-view image information, or if no such data is available, by a point rasterization step. Finally, all polygons are combined to a joint, textured 3D model.

4.1.1 Input Data

Our reconstruction system uses a 3D point cloud and optionally photos with known camera parameters (intrinsic parameters, position, and orientation) as input data. The points have to be oriented, i.e. each point has a surface normal assigned. If surface normals are not available from the input source, it is possible to estimate the normals with a simple least squares method. This estimation produces smooth normal changes at sharp edges, but in our approach exact object edges are extracted from the input images or computed by intersections.

The input data can be acquired by image-based *structure from motion* techniques [44, 136], or by laser scanners with additional, registered photos. While both capturing methods provide a good impression of the scene to reconstruct, common artifacts like noise, mismatches and holes cannot be avoided due to reflective surfaces, occluders or varying lighting conditions.

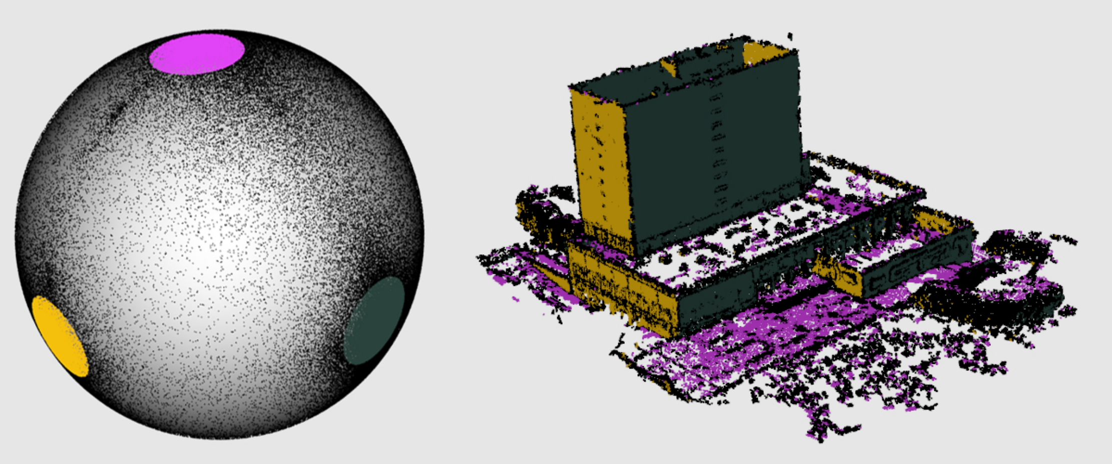


Figure 4.2: Three orthogonal main axes have been identified in this scene. The colored areas on the normal sphere (left image) denote all points belonging to one axis cluster. The clustered points are shown with the same colors in the 3D view (right image).

4.2 Point Cloud Segmentation

The goal of our point cloud segmentation is to divide the oriented points into planar pieces and to remove points which are not located on a planar surface. The point cloud is first divided into clusters with similar surface normals and subsequently according to spatial distances. Finally, all unclustered points are evaluated and potentially added to an existing cluster. The segmentation is performed iteratively on the remaining points until no additional clusters can be found. Usually not all points are assigned to cluster, because the input cloud contains outliers or non-planar areas.

4.2.1 Axes Search

The segmentation process starts with the search for dominant axes in the scene by evaluating all surface normals. For typical urban scenes the segmentation is improved by fitting predefined axis frames ($90\,^\circ$, $45\,^\circ$ or $60\,^\circ$) to the surface normals: Inaccuracies from the normal estimation are removed by snapping the orientation to predefined axis frames. Furthermore, well aligned point clusters are also detected when they only contain a relatively small amount of points, because they are related to larger clusters. For our automatic pipeline we search for 45 degrees in the first iteration, and then for 60 degrees in the remaining points. Finally we search for other dominant axes that are not part of predefined axis frames.

The axis frame fitting uses a RANSAC approach. Two surface normals are taken randomly from the point set and define the orientation of the selected axis frame. The quality of a specific axis frame orientation is defined by the number of inliers, i.e. the surface normals that are within a maximum angle α to one of the predefined axes. Axes which have only a small number of inliers are removed from the axis frame. Figure 4.2 shows an example for a fitted orthogonal axis frame and the corresponding inlier normals.

The orientation of the best axis frame is improved by an iterative mean-shift [24] optimization step as denoted in Equation 4.1, where a identifies an axis direction, n a normal and *Rot(u,v)* returns the rotation matrix between two directions. For each axis, the mean unit vector is calculated from all normals within angle α weighted with function *w*. For each axis, the rotation to its local maximum on the unit sphere is computed. Finally, a global transformation is computed for all axes as the weighted average of all rotation matrices, followed by an SVD-based orthogonalization step. The weights are proportional to the number of points assigned to each axis. The final transformation is applied to all axes, and the procedure is iterated until convergence or up to a maximum number of iterations.

$$\begin{aligned} R &= \sum_a \mathit{Rot}\left(a, \frac{\sum_n w(a,n)\cdot n}{\|\sum_n w(a,n)\cdot n\|}\right)\cdot \sum_n w(a,n) \\ w(a,n) &= (\frac{n\cdot a-\cos(\alpha)}{1-\cos(\alpha)})^2 \end{aligned} \tag{4.1}$$

The search for axes without angular relations starts with equally spaced seed axes on the unit sphere. The number of seeds depends on the angle α, which defines the maximum distance for potential inliers. All initial axes are independently optimized by mean-shift rotations, and will converge to centroids of local clusters. Only axes with sufficiently large clusters at their final position are accepted.

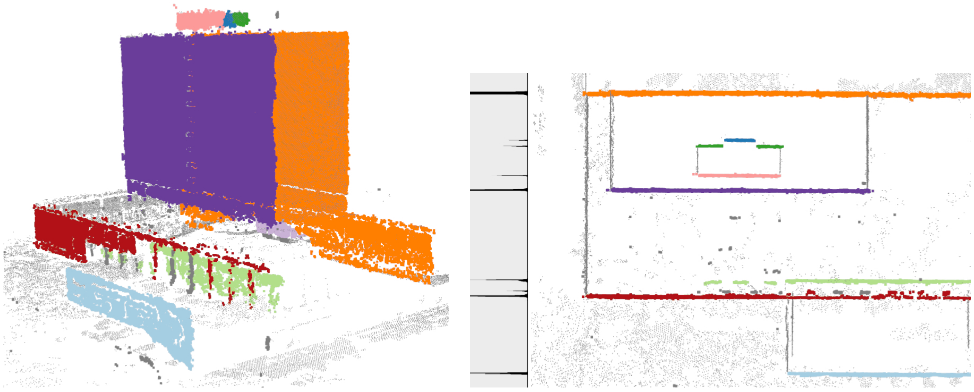
4.2.2 Planar Segmentation

In the previous section we have segmented the point cloud into areas with a common surface orientation. The next step is to divide these clusters into planar segments. Points with orientation close to a detected axis direction are grouped to a cluster, wherein multiple 3D planes are fitted. All cluster points are projected to a 1D space along the axis direction and clustered in order to find dominant planes (see Figure 4.3). Detected planes are fit to their cluster points with weighted least squares. A small threshold is needed for clustering along the axis orientation in order to prevent two planes from merging.

Subsequent to the plane search, a 2D distance based clustering is performed to separate different surfaces which are located in the same plane. The points are clustered with single-linkage, i.e. there is a minimum distance between any two points of different clusters. The minimum distance threshold can be chosen less strictly, since potential holes in a surface will be removed during polygonization.

4.2.3 Point Evaluation

The previous clustering steps might miss some points due to noisy 3D positions or erroneous normals. Therefore, the final step of one iteration evaluates each unassigned point and adds it to the best-suited point cluster within a maximum distance. Similar as in Section 4.2.2, a strict threshold is used for the orthogonal distance to the plane, while a loose threshold is used for the



(a) (b)

Figure 4.3: Clustering points along an axis direction for detecting planes: The left image shows the 3D view of a scene where points with the same orientation have been segmented into planar regions. The right image shows the top view of the scene, and a histogram that shows the distribution of points along the cluster orientation. As can be seen, the detected planes correspond to peaks in the point distribution along the cluster orientation.

distance to the nearest point in the cluster. The remaining unclustered points are processed in the next iteration of the point cloud segmentation. Additional iterations ensure that smaller areas with different axis orientations are detected as well.

4.3 Polygonization

The final reconstruction step generates triangle meshes for the planar point clusters. The points are usually densely distributed, but the segment borders are not exactly defined, and holes may occur due to missing data. We provide two different algorithms for creating polygons: the first one, called point rasterization, only depends on the point cloud itself, while the second one uses image information and consists of three substeps (an overview is given in Figure 4.1).

4.3.1 Point Rasterization

Each point cluster is transformed into 2D space by projecting it onto its fitting plane. The 2D points are rasterized into a binary image, and small holes are removed by a morphological close operation (cf. Figure 4.4). The resolution of the binary image is either predefined by the user or it approximately corresponds to the average distance between input points. In order to avoid rasterization artifacts, the points are aligned to globally dominant axes before rasterization. These dominant axes can be taken from the calculated main axes for normal segmentation (Section 4.2.1) or computed by principal component analysis.

This approach is fast and has proven to be robust in our test scenes, but artifacts in the point cloud propagate to the final polygon mesh. It is not possible to discriminate between real surface

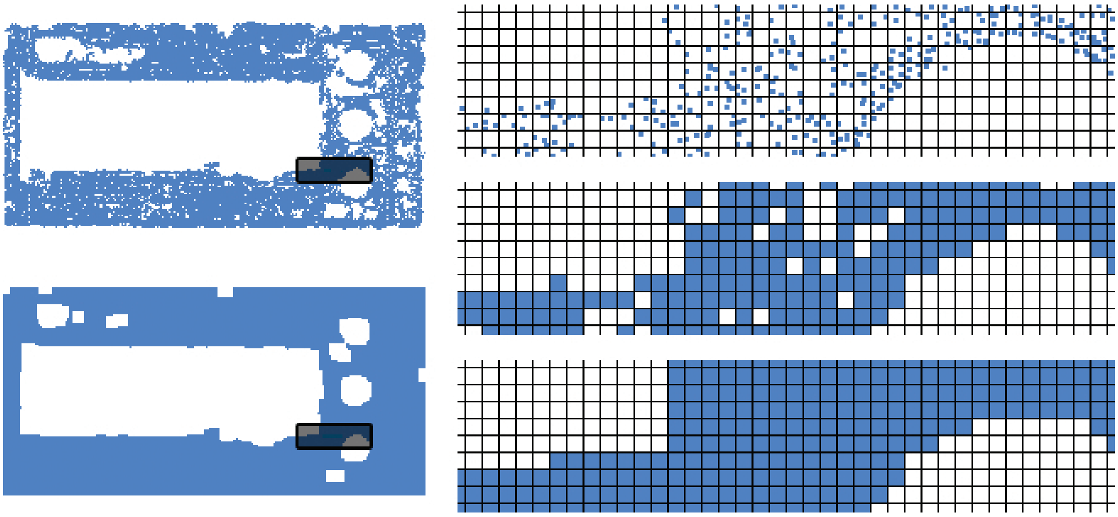


Figure 4.4: Cluster points are rasterized into a binary image. Morphological closing is applied to remove small holes.

holes and artifacts due to missing data, and surface boundaries are often jagged. Hence, we incorporate image information as described in the next sections. Nevertheless, the rasterization approach is a good alternative for areas with missing or inferior image information.

4.3.2 Image Projection

The image-based surface reconstruction starts with selecting the best suited images for each planar cluster. In our implementation, we automatically select the four best images from the input data. The selection favors photos where the camera plane is nearly parallel to the fitted plane. Additionally, the planar point cluster should be projected onto a large image area without being occluded by other objects. In practice, this is done with the help of hardware occlusion queries and point splats in order to test if a point cluster is occluded by other parts of the input point cloud.

The selected images are projected onto the slightly increased bounding region generated by the cluster points in the fitted plane. The projected images overlap correctly in areas where the plane coincides with the real surface. The goal is to identify the correctly projected areas for defining the correct boundaries of the plane. Because buildings often contain large homogenous areas, it is not possible to directly analyze the projection areas and instead we find the correct plane boundaries along strong image edges.

4.3.3 Edge Detection

The final polygons should be aligned with strong image edges which correspond to edges located in the cluster plane. Gradient images are generated from all projected images, but high gradient values are only accepted if they occur in a majority of the images. This prevents taking edges

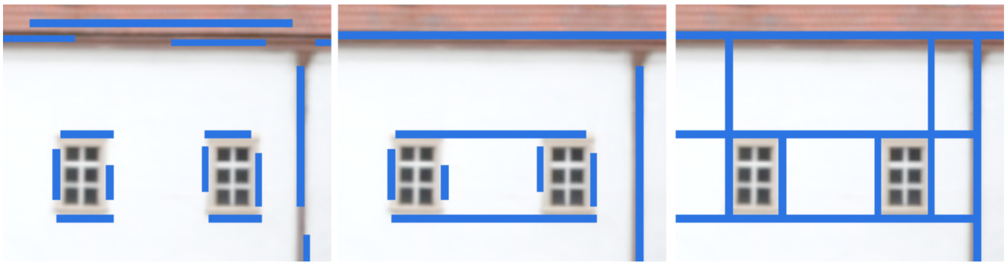


Figure 4.5: Nearly collinear line segments are merged (center image) and subsequently connected to form a grid (right image). Each line segment is elongated up to its nearest intersecting line.

into account which are not located on the current plane surface. Computing the gradients in the individual images is robust to small registration and fitting errors. Blending all input images would smooth strong features and decrease the discriminative power of large gradient values.

Image edges are not only evoked by depth changes, but also to a great extent by changes in texture which are not relevant to our application. Another observation is that in the case of buildings, depth changes are often aligned to orthogonal main directions. For this reason, we restrict image edges along two main directions. Straight lines are extracted from the cumulated gradient image with a Canny edge filter followed by an edge linking step. Two perpendicular main directions are extracted from global directions or computed in the current plane image by clustering, and only edges along these directions are accepted.

4.3.4 Grid Filling

The set of line segments is simplified by merging nearly collinear segments. Based on the resulting line segments, a rectilinear grid is created by extending all lines to their next intersection with another line (cf. Figure 4.5). The outer grid boundary is defined by the union of all line segments and all projected cluster points.

Finally, we have to select the grid cells that are part of the 3D model. The point density for each cell is defined as the ratio between the number of its inlying points and its area, normalized by the median density of all cells. Cells belonging to the object surface are identified by thresholding their density. A polygon per cluster is then created by simply transforming the boundaries of accepted cells from 2D plane coordinates to 3D world coordinates.

4.3.5 Combination and Texturing of Polygons

In the previous sections the polygonization was performed for each point cluster separately, but in most cases the individual polygons have direct neighbors from other point clusters. The overall result can be highly improved by closing gaps and correcting intersections between neighboring polygons (see Figure 4.7).

Clipping splits the geometry of a cluster by the intersection line of the two neighboring polygon planes. A part is deleted if its relative size is very small, or if the number of inlying cluster

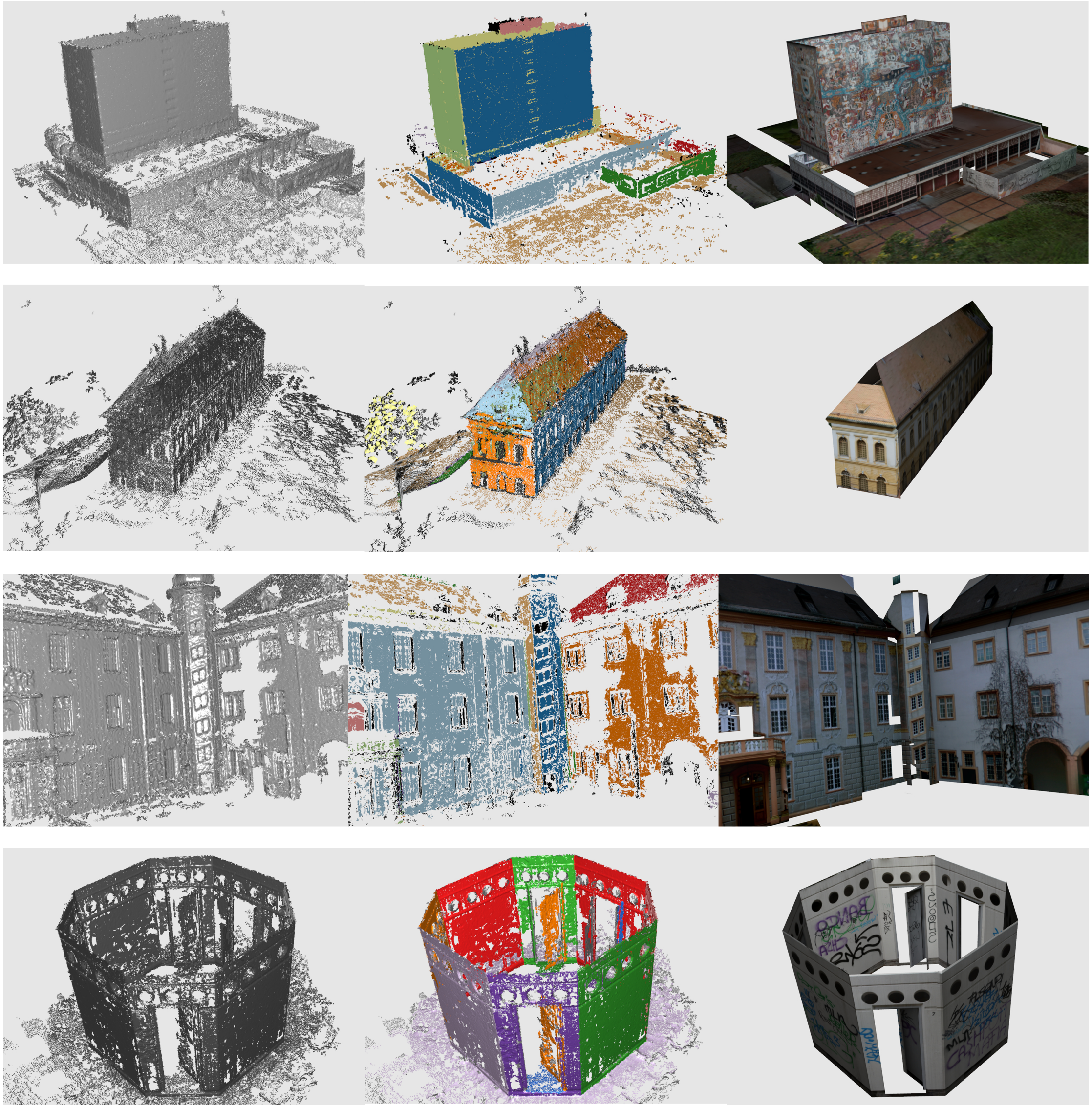


Figure 4.6: Results (from top to bottom): Library, Castle-P19, Manor and Octagon.

points is very low. *Merging* extends the existing geometry from the polygon border to the plane intersection line if the relative size of the new part is small. Both operations have very low complexity since all polygons are planar and have a low number of triangles.

The combined polygonal 3D model is textured by stitching together the photos selected in the image projection step (Section 4.3.2). For each pixel in the final texture image, the best image shot is selected based on distance and angle between scene plane and camera. Poisson image blending is used for combining multiple input images without visible seams [94, 106].

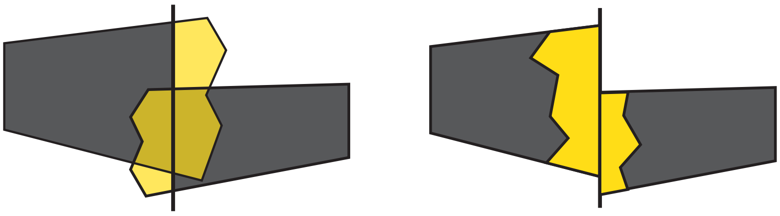


Figure 4.7: Polygon Combination: Nearby polygons are connected by clipping (left) or merging (right).

	Points	Images	Segments	Time (s)
Library	253126	21	19	91
Castle-P19	330165	19	23	105
Manor	179278	21	10	68
Octagon	560092	27	19	82

Table 4.1: Detailed numbers about the data sets in Figure 4.6.

4.4 Results

We have tested our approach on several input data sets. All point clouds have been created with PMVS [45] from oriented images. Figure 4.6 shows the input point cloud, the segmentations and the final 3D model for each test data set. The second data set *Castle-P19* is a publicly available multi-view test data set [140]. Table 4.1 lists the number of input points, input images, planar segments, and the computation time for generating the triangle meshes.

4.5 Conclusions

We have presented a fully automatic pipeline for generating low-poly 3d models of buildings from a point cloud and additional information from oriented images. The method is divided into a planar segmentation of the point cloud, and a subsequent polygonization of the point segments.

Due to the coarse approximation of buildings as planar polygons, small features are not reconstructed in detail. This is largely compensated by the generated textures, and can be additionally enhanced by advanced texturing techniques such as normal or displacement mapping. Moreover, the low geometric complexity might be desired in various applications, or a starting point for further manual modeling.

While the algorithm provides convincing results for buildings, it is not suited for other input scenes such as building interiors. For such complicated input scenes it is often not possible to automatically generate 3D models of high quality. Therefore, it is desirable to combine the algorithm with interactive user input in order to improve the results and to enlarge the set of possible input scenes.

CHAPTER 5

Interactive Reconstruction from RGB-D Images

In this chapter, we present a system for reconstructing surfaces from both, images and point clouds. We focus on input from RGB-D cameras, which capture a scene with multiple color images as well as dense point clouds, but many parts of this reconstruction algorithm can be applied to input data from other sources as well. Similar to Chapter 3, the reconstruction is based on multiple shape primitives, which provide a higher level of abstraction for effectively using, analyzing or editing 3D data.

Similar to Chapter 4 we start with the segmentation of a point cloud, but in contrast we support multiple types of shape primitives, namely planes, cylinders, and cones. The main contribution of this algorithm is that we efficiently handle the redundancy provided by multiple RGB-D images (Section 5.1). If unstructured point sets are used, other methods can be used for segmention, e.g. the RANSAC-approach presented by Schnabel et al. [127].

The second part of our reconstruction algorithm extracts and optimizes boundaries for the detected shape primitives. The optimization framework works for a large set of possible constraints, which can be either automatically computed or provided by user interaction. While it is still possible to fully automatically generate 3D models, an intuitive user interface is presented for completing and improving the automatic results. In Section 5.2 we describe our novel optimization framework, and in Section 5.3 we present possible constraints.

5.1 Segmentation

In this section we present a novel method for segmenting the 3D point cloud into shape primitives. Currently, we support planes, cylinders, and cones as shape primitives. Figure 5.1 shows an example of the segmentation algorithm and provides an overview of the individual steps of the algorithm. Besides surface reconstruction, the generated segmentation might be used for dividing large scenes into smaller parts for simplifying further processing.

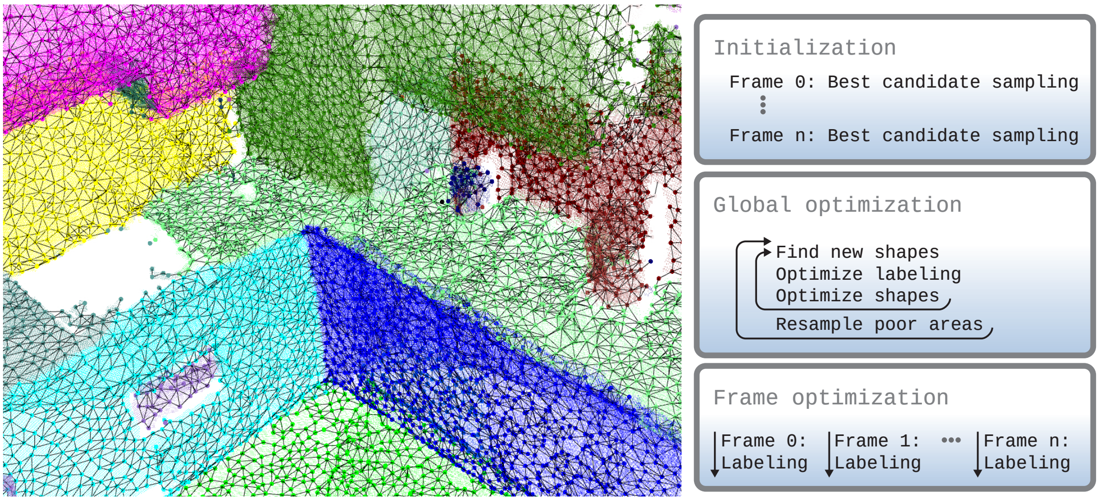


Figure 5.1: Segmenting multiple depth images of an interior scene into geometric primitives: The example on the left side shows, how the segmentation is divided into a global graph-based segmentation (shown by large points and edges) and into a pixel-based segmentation of the individual depth images. An overview of the segmentation algorithm is given in the right image.

Our segmentation technique uses multiple registered depth images with known intrinsic and extrinsic camera parameters as input. All pixels with a valid depth measurement can be transformed to a 3D position. While depth images can be easily converted to 3D point clouds, they provide additional useful information. The organization in 2D grids implicitly provides connectivity information. The precision of depth measurements usually decreases with the distance to the camera. This knowledge allows to use different distance thresholds across a scene for handling accurately defined as well as noisy parts at the same time.

The input depth images (also called frames) are registered to a common coordinate system. We use color features and iterative closest points (ICP) [56], and improve the input data by a denoising algorithm [114]. The registered frames can be interpreted as a large graph of 3D points, in which neighboring pixels within one frame as well as corresponding points of different frames are connected by graph edges. This neighborhood information could be used for computing a globally consistent segmentation of all input frames. Unfortunately, the data from multiple frames would be too large for such a brute-force approach. Therefore, we divide the problem into two parts: First, a globally consistent segmentation is computed for sample points covering the whole scene. Second, a pixel-accurate segmentation is computed individually for each frame based on the global segmentation (see Figure 5.7). In summary, the contributions of this section are:

- Multiple range images are converted into a three-dimensional graph structure where the provided connectivity information of 2D depth maps is exploited.
- The 3D graph is segmented into shape primitives. A novel iterative approach enables the detection of shapes and a consistent labeling across the scene.

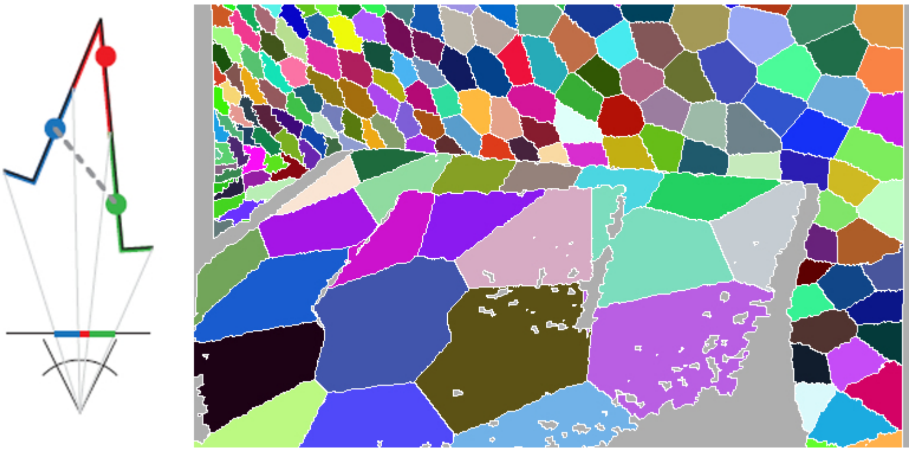


Figure 5.2: Left: The connectivity of depth maps is transferred to the 3D graph. Although the green and blue node are close in 3D, they are not connected because their areas do not meet in the depth map. Right: Pixels are assigned to their closest graph node for extracting neighborhood relations.

- The global segmentation is projected and optimized for individual range images.

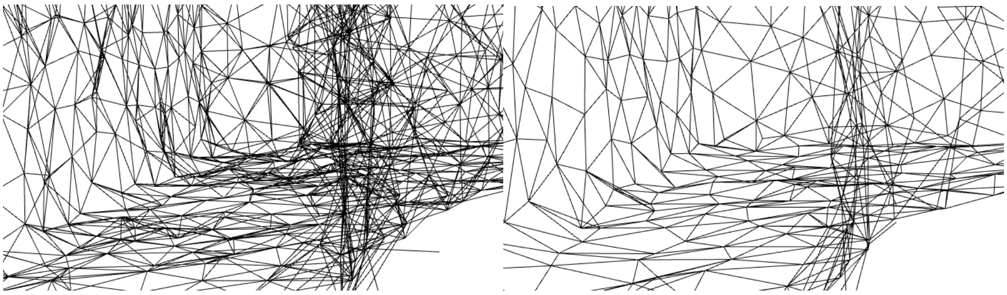
5.1.1 Graph setup

The central data structure for our algorithm is a sample point graph which is formed from a subset of 3D points sampled from all pixels in the input frames. The goal is to reduce the number of 3D points for efficient segmentation, and to project corresponding pixels from different input frames to the same 3D point.

Whenever a new frame is added, the sample point graph has to be updated to include newly captured areas. The point cloud is sampled such that the average distance between neighboring points is approximately τ. This parameter adjusts the number of nodes in the graph and provides a trade-off between accuracy and computational efficiency.

We use the modified best-candidate sampling algorithm provided by [21] for selecting new samples. Random sample points from the current frame are generated iteratively and the best candidate (with the largest distance to existing sample points) is selected. The average distance of the best candidates is kept over the last 100 trials and adding points is stopped after it falls below τ. For each frame it is only necessary to consider existing graph points that are projected onto the current frame or within the maximally projected length of τ.

Graph nodes should be connected to neighboring nodes when they are close and when they are located on the same surface of an object. The latter condition would require to compute geodesic distances between 3D positions, which is not possible for unstructured point clouds. Nevertheless, it is possible to use the connectivity information given by the 2D-alignment of range images. Potential neighbors in the sample point graph are nodes with a distance below 2τ. While this value is quite large, incorrect neighbors will be pruned by the following algorithm: Each pixel in the current input frame is assigned to its nearest sample point in 3D space (right



(a) Initial sampling (b) With adaptation

Figure 5.3: Adapting sample points to multiple frames highly reduces noise in the sample point graph.

image in Figure 5.2)). Potential edges between two nodes are only accepted if at least two neighboring pixels are assigned to the according sample points. This approach prevents connecting close points of different surfaces, e.g. at sharp edges as illustrated in Figure 5.2. The 3D points of all assigned pixels are also used for estimating the graph node's orientation.

In order to reduce noise in the sample point graph, we adapt positions and normals when an existing sample point is visible in a new frame. We compute the weighted average between an existing node and corresponding points in the new frame, though points can only move along their normal. The weights are inversely proportional to the camera depths, and the weights for new points are decreased with every considered frame. Thus, we are pushing sample points towards precisely measured points. Figure 5.3 shows how noise is removed from the graph and the number of nodes and edges is reduced.

5.1.2 Global segmentation

The global segmentation uses only the sample point graph and is completely independent from the source images. The goal is to find primitive shapes in the graph positions and assign each node to the best shape. The segmentation is divided into three parts, namely detecting new shapes, assigning graph nodes to shapes (labeling), and optimizing shapes, which are repeated until convergence.

Detecting shapes We use a RANSAC approach [127] for detecting primitive shapes in all graph positions which are not yet assigned to a shape. In the beginning, this will include all graph positions. The parameter ϵ defines the maximum distance between shapes and points, and should be set according to the expected amount of noise in the input data.

Depth measurements of stereo systems produce systematic errors proportional to the distance squared. In order to account for these varying noise levels across range images, the expected measurement error at a specific range image pixel is added to the distance threshold ϵ. For Kinect sensors we take the accuracy measurements provided by Khoshelham and Elberink [71] and compute the expected measurement error as $2.85e-6\text{dist}^2$ for measurements in millimeters.

Labeling Assigning graph nodes to shapes can be formulated as a labeling problem, which we solve with graph cuts [15]. The optimization objective is defined as $E_{data} + \lambda E_{smooth}$, where the first term optimizes the distances between graph samples and shapes and the second term strives for assigning neighboring graph nodes to the same shape.

Graph nodes should be assigned to a shape, if the distance to the surface is small and if the point normal coincides with the surface normal, which leads to the following data term:

$$E_{data} = \sum_{i \in G} \frac{d(p_i, S_{l(i)})^2}{3\epsilon} (1 - (n_i \cdot n(S_{l(i)}, p_i))) \tag{5.1}$$

G contains all graph nodes, p_i and n_i denote positions and normals, $l(i)$ is the index of the assigned shape S. The function d computes the distance between a point and its closest point on a shape surface, the function n retrieves the surface normal at the closest point on the shape.

In order to handle outliers and scene parts that cannot be modelled by primitive shapes, there is an additional unassigned-label for points far from all detected shapes. A constant cost value is applied to such unassigned nodes. A low value might lead to large areas of unassigned points, while a high value might pollute shapes with outliers. In our experiments we generally use a value of 0.8 as cost for unassigned nodes.

The smoothness term strives for assigning nearby points to the same shape. It is defined as

$$E_{smooth} = \sum_{(i,j) \in E} \delta(l(i) \neq l(j))(n_i \cdot n_j), \tag{5.2}$$

where E is the set of all edges in the sample point graph and the function δ denotes the Potts model, which returns 1 if the argument is true and 0 otherwise. All edges are weighted based on their normal deviation. If two nearby points have very different orientations, it is more likely that they belong to different shapes.

Shape Optimization After the set of assigned points has changed, a refitting step is applied. The geometric error of the shapes is optimized with weighted least-squares. Shapes which have no or only very few points assigned are deleted and the points are labeled as unassigned.

Some scene parts might be slightly distorted in the point cloud, especially when uncalibrated cameras are used and the scene is captured only from few view points. This might lead to wrong initial shape types, e.g. a planar wall is represented by a small part of a cylinder with a large radius. We use a simple heuristic, where all shapes are tested whether they can be replaced by a plane without increasing the average residual more than twice. Cones might also be replaced by a cylinder.

Resampling During initialization, the sample point graph is generated with a fixed value of τ, which defines the average distance between graph nodes. For some scenes it is useful to apply different values of τ to different parts. For example, walls of an interior room can be modeled with a low number of graph nodes, while more nodes are required for furniture and other objects.

For this reason, it is optionally possible to restart the segmentation process with a smaller value of τ after the algorithm has converged. Areas which are not well described by the current

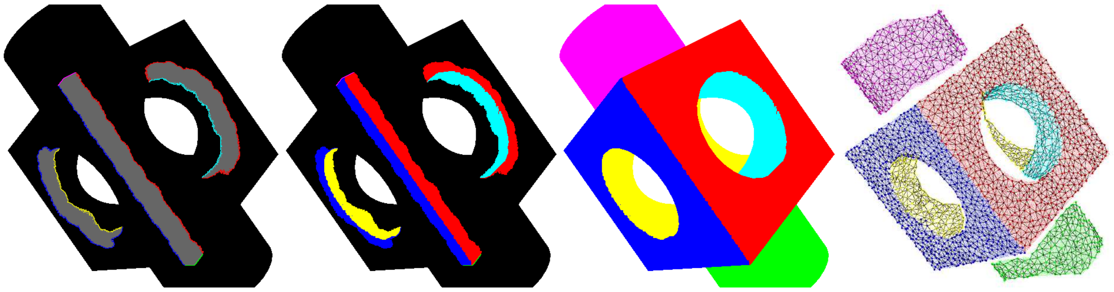

(a) (b) (c) (d)

Figure 5.4: Frame optimization (white pixels do not have depth values, black pixels are not included in optimization): (a) Dilated pixels with fixed labels, (b) optimized pixels, (c) all pixels (optimized and directly used from global graph), and (d) 3D view.

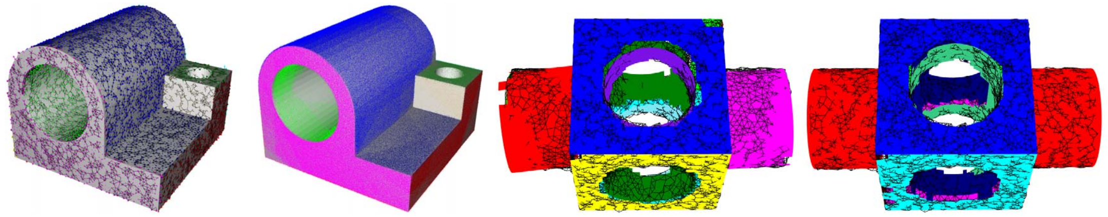
segmentation, i.e. areas with unassigned graph nodes, are resampled with a smaller τ. The according graph nodes are replaced by denser samples.

5.1.3 Frame segmentation

The final step of our segmentation algorithm is to project the segmentation of the global graph to the individual frames. The segmentation is computed for each frame individually, and has to be consistent with the global segmentation. Only pixels with a valid depth value are considered because 3D positions are needed. For each pixel, the graph positions within a distance of 3τ are investigated. If all have the same label and the pixels 3D position and normal are consistent with the according shape, the pixel is labeled with this shape. Otherwise the pixel is marked as unsolved, and the best label has to be determined by optimization.

We apply a graph cut optimization for all unsolved image regions. The regions are enlarged by dilation to ensure valid transitions to the rest of the image. A unique label has already been determined for these dilated pixels. Thus, we apply a cost value of zero to this label and a very high cost value for all other labels. For the unsolved pixels the same optimization objectives are used as for the global graph optimization (Eq. 5.1 and 5.2). The smoothness cost is applied to neighboring pixels if they are not divided by large depth discontinuities. Additionally, we want to ensure that the labeling of the global graph is exactly taken over to the individual frames. Therefore, an additional penalty is added to the data costs for labels not present in the neighborhood of a pixel. Figure 5.4 shows several stages of optimizing a frame.

Each frame is handled individually and is completely independent from other frames. This approach is memory-efficient and multiple frames can be easily processed in parallel. Depending on the application, it might be a problem that small overlapping areas of different frames are inconsistently labeled when there are no sharp features.



(a) Joint (b) Block

Figure 5.5: (a) Global graph and frame segmentation of virtually created depth maps of the joint model. (b) Comparison between the initally detected shapes and result of our optimization on the block model. Both models are courtesy of AIM@SHAPE.

	Kitchen	Block	Joint
Frames	104	20	30
Points/Frame	253777	133387	113271
Nodes	22809	6323	14104
Shapes	31	11	11
Init/Frame	0.637	0.256	0.365
Global Seg.	275.2	3.270	7.708
Seg/Frame	8.101	2.925	1.948

Table 5.1: Runtime measurements on processed models, executed on an Intel Core i7, 2.67 GHz CPU. All timings are in units of seconds. The frames have a resolution of 640x480 pixels, but not all pixels have a valid depth value.

5.1.4 Results

We have tested our algorithm on different input scenes. The results show that basic shapes are successfully detected and the depth maps are segmented along object boundaries. Figure 5.5 shows results on virtually created depth maps. Figure 5.5(b) shows how our new optimization can improve the initially detected shapes. In this case, especially the shape parameters of the left horizontal cylinder have been improved and the left and right cylinder have been merged.

Real world examples can be seen in Figure 5.6. The kitchen and the office scene have been segmented with a fixed value of τ, while for the sofa scene three different values of τ have been used. It can be seen, that the sampling of walls is not as dense as the sampling of the furniture. Table 5.1 shows timings of our algorithm. The frame optimization is the most time-consuming part of our algorithm, but for some applications the global optimization alone might be sufficient. In fact, frame segmentations are not necessary for the following reconstruction algorithm when accurate shape boundaries are extracted from other input sources.

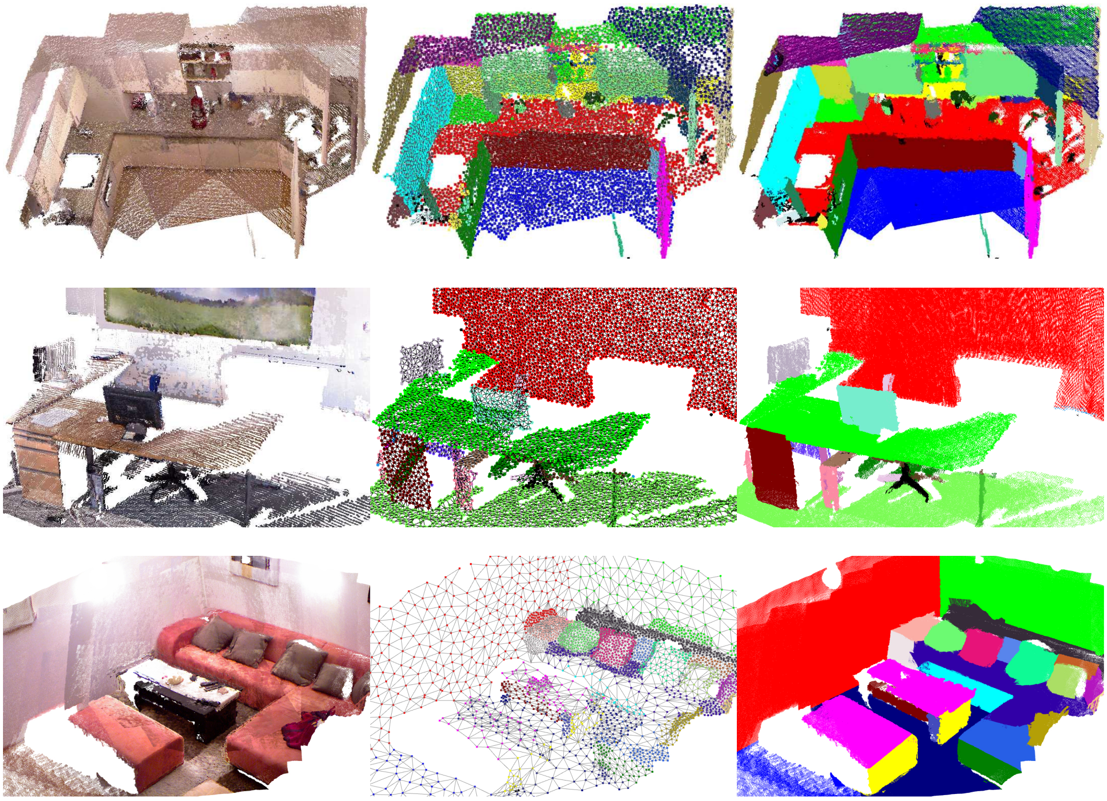


Figure 5.6: Segmentation of real-world scenes (kitchen, office, and sofa). From left to right: colored input frames, segmentation of sample point graph, and all segmented frames.

5.2 Boundary Optimization

In the previous section we have segmented a point set into multiple primitive shapes. Many shapes, e.g. planes or cylinders, have an infinite extent in one or more dimensions. Also, surfaces do not always occupy the full extent along finite dimensions. Thus, generating a valid 3D model requires an additional step to extract the boundaries of the shapes. Most algorithms so far only deal with the detection of shapes, and create the final 3D model by intersecting nearby shapes [82], or combining all shapes to a closed model [126]. These approaches may fail if a scene is only partially reconstructed, contains large holes, or some shapes have been wrongly detected due to noise. In case of thin objects, such as table tops or doors, it is usually not possible to robustly estimate the thin sides just from a noisy point cloud. Especially in interior scenes, point clouds can only be captured from a limited space of viewpoints. Hence, only the front faces of an object are well defined by the point cloud.

The primary goal of our optimization framework is to find a good surface boundary for a shape primitive, i.e. dividing the shape into *inside* and *outside* areas. This task is very similar to image segmentation techniques, where an image is divided into foreground and background. In-

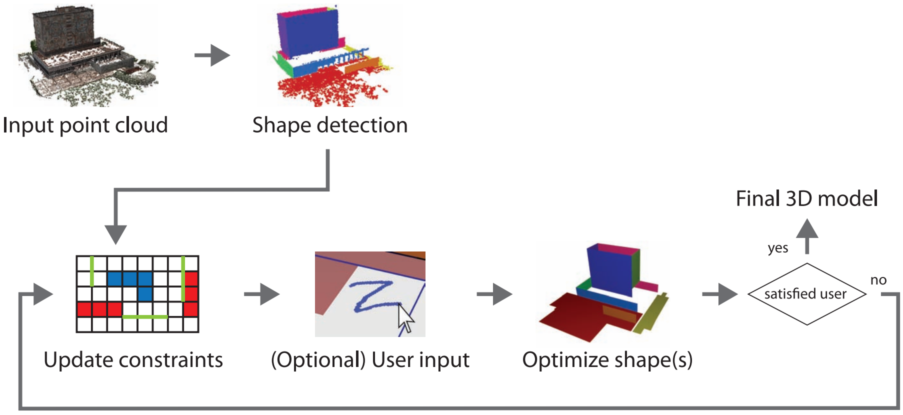


Figure 5.7: Overview of our reconstruction pipeline.

spired by popular image segmentation algorithms [15], we solve the problem of shape boundary optimization with graph cuts.

We parameterize the surface of a three-dimensional shape in a two-dimensional space and uniformly rasterize it into a set of pixels I. The binary segmentation of the 2D space is obtained by minimizing the cost function defined in Eq. 5.3, containing a regional and a boundary term. Energy minimization is performed with the α-expansion algorithm [16]. Compared to the simple rasterization presented in Chapter 4, the binary segmentation is not limited to 3D point sets as input.

$$E = \sum_{p \in I} R(p) + \sum_{(p,q) \in N} B(p,q) \cdot \delta(A(p) \neq A(q)) \tag{5.3}$$

The result of the minimization is a labeling $A(p)$, where each pixel p is marked either *inside* or *outside*. $p \in I$ denote all pixels in the 2D shape parameterization, N contains all pairs of neighboring pixels under a standard 4-neighborhood system. The function δ denotes the Potts model, which returns 1 if the argument is true and 0 otherwise.

The regional term $R(p)$ provides different penalties on labeling a pixel inside(R_{in}) or outside(R_{out}). The boundary term $B(p,q)$ puts a penalty on a discontinuity between the pixels p and q, i.e. when they are assigned to different labels (see Section 5.2.1 for more details).

The optimization is applied only on a limited space of the shape defined by an oriented bounding box. The boundary will be wrongly clipped if the bounding box is selected too small, while a large bounding box will increase the computational time without cause. In our experiments, we found a good compromise by initializing it as the bounding box of the initial point cloud assigned to a shape, enlarged by 20%. After each optimization step, the bounding box is extended according to the new shape boundary. The size can also be manually increased in our interactive user interface, e.g. if the default does not work due to large amount of missing data in the input point cloud.

Each shape is optimized separately, but many constraints are computed based on information from the whole model. For example, intersections are computed between pairs of shapes and each 3D intersection curve is projected onto two shapes. Thus, the shapes are optimized locally under global constraints. Some constraints depend on the current shape boundary, e.g. the coplanarity constraint tries to find dominant planes in all shape boundaries. Therefore, multiple optimization steps are executed, interleaved with an update of all constraints, as illustrated in Figure 5.7. This ensures that local changes are propagated back to dependent constraints and subsequently to other shapes.

5.2.1 Constraints

A wide range of input sources can be used as constraints for the boundary optimization. For example, the final polygon should contain all points used for estimating the shape, the boundary should match edges in images, or parallel lines are preferred (see Section 5.3 for detailed information). These constraints either influence the regional term $R(p)$ or the boundary term $B(p, q)$ of Equation 5.3.

Regional constraints define which pixels should be labeled as inside respectively as outside. They provide two grayscale images, which correspond to the costs for labeling pixels as inside(R_{in}) or as outside(R_{out}).

Boundary constraints provide good locations for the shape boundary, i.e. for changing between inside and outside. These locations are stored as a set of line segments. The line segments are rasterized for the optimization, but their accuracy is independent from the shape resolution, which is essential for avoiding rasterization artifacts (see Section 5.2.2).

Multiple constraints can be used at the same time by summing up the individual costs. Using multiple constraints leads to conflicts when they vote for different labels. For this reason it is necessary to make some constraints more important than others. These conflicts can be easily solved by weighting the constraints with different impact factors which are editable by the user. The relative differences between impact factors are the important information, not the absolute values. For example, constraints from user input are always weighted higher than other constraints.

5.2.2 Rasterization artifacts

The pixel-based approach introduces the problem of rasterization artifacts. The polygon positions are aligned along the main directions of the rasterization grid and hence they are not exactly positioned on the constraint lines. For scenes with planar surfaces a lot of artifacts can be removed by aligning the local coordinate systems on planes with the major directions of the scene. Also, a higher resolution would reduce the rasterization errors at the cost of higher computation times. We propose a constraint snapping algorithm for generating exact boundaries for all types of shapes which works well also under a low optimization resolution (see Figure 5.8).

The original line segments from all boundary constraints, which generally have a higher precision than the rasterization grid, are taken as input. The polygon created from the rasterization

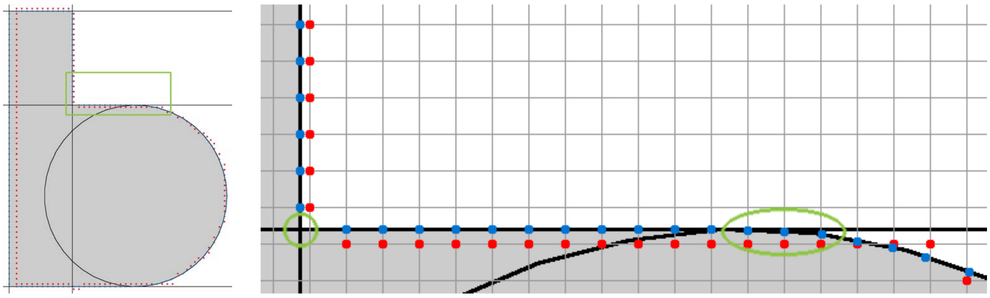


Figure 5.8: Snapping the computed boundary to original constraint lines. The original boundary is located on the rasterization grid (red sample points). The final boundary is snapped to nearby constraint lines (blue sample points). On the left side (green circle) a corner point will be inserted, while at the right side (green ellipse) a smooth transition between the constraints is achieved.

grid is sampled with the desired output tesselation. The sampling distance defines the precision of the final polygons and has to be smaller than the tesselation of curved objects. A graph cut optimization is applied for aligning the sample points with nearby constraint lines.

The cost for assigning a sample point p to a constraint line c is defined in Equation 5.4. It is based on the distance between the point and line $\operatorname{dist}(p, c)$, the constraint line's weight w_c and the angle between the direction of the polygon at point p (dir_p) and the constraint line direction (dir_c). This has the effect that points are snapped to nearby lines with corresponding directions and that lines with higher weights are preferred. For sample points that are not snapped to a constraint line, we choose a constant cost value of 2.25. This means, that constraint lines with highest weight and perfect direction will attract points at a maximum distance of 1.5 pixels.

$$E(p, c) = \operatorname{dist}(p, c)^2 + \exp\left(\frac{-w_c^2}{(w_{max}/4)^2}\right) + (1 - \langle dir_p, dir_c \rangle) \tag{5.4}$$

Neighboring sample points are preferably assigned to the same constraint line to avoid oscillation between two nearby constraint lines. Additionally, the transition between two different constraint lines should be located where the constraint lines have the smallest distance to each other. Equation 5.5 shows the cost function for assigning point p_i to constraint c_k and point p_j to c_l. The function $\operatorname{clp}(p, c)$ returns the closest point on line c to point p.

$$\begin{aligned} E(p_i, p_j, c_k, c_l) = & \parallel \operatorname{clp}(p_i, c_k) - \operatorname{clp}(p_i, c_l) \parallel \\ & + \parallel \operatorname{clp}(p_j, c_k) - \operatorname{clp}(p_j, c_l) \parallel \end{aligned} \tag{5.5}$$

The optimization snaps points to nearby constraint lines and provides smooth transitions between different constraints. Corners are extracted from intersections between constraint lines or neighboring lines with sharp angles. The corners are inserted into the final polygon between the closest points on the according constraint lines.

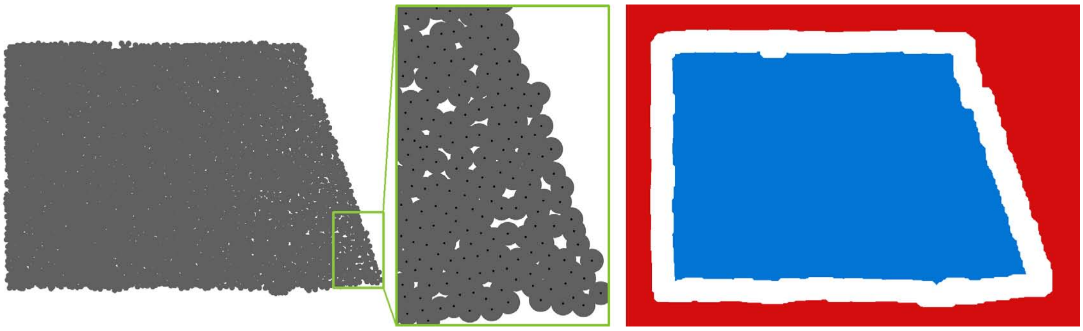


Figure 5.9: Left: 3D points are projected onto the 2D parameterization of a shape. Neighboring points are connected due to the automatically computed point size as can be seen in the detailed view. Right: The costs based on the projected points are visualized. Blue pixels should be labeled inside (high value for R_{out}) and the red pixels outside (high value for R_{in}). The boundary will be located within the white area.

5.3 Constraints

In this section we present important constraints that can be used by our optimization framework. Constraints are either generated from input data such as point clouds or images or they are derived from the shapes themselves or from the current shape boundaries.

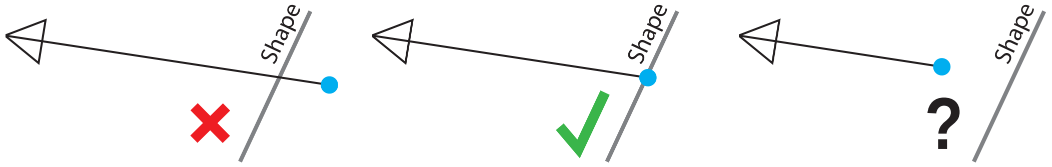
Generally, the input data for detecting shapes is included into the optimization. This is either a point cloud (Sec. 5.3.1) or a set of oriented depth maps (Sec. 5.3.2), which additionally provide information on occlusions over point clouds. Intersection constraints (Sec. 5.3.3) are important for generating closed meshes where the individual shape boundaries exactly meet each other. Optionally, edges from color images can be included, when images are available and contain visible object boundaries (Sec. 5.3.4). We present two constraints, which analyze the current shape boundaries for coplanar segments (Sec. 5.3.5) respectively for dominant two-dimensional shapes (Sec. 5.3.6). These constraints are often combined with manual inputs (Sec. 5.3.7), where the user pushes the boundary towards the favored shape which is then automatically refined by other constraints.

In our system individual constraints provide costs in the range of zero to one. This guiding principle facilitates the task of assigning different impact factors to constraints in order to favor important constraints over others (see Section 5.2.1).

5.3.1 Point Cloud

In our reconstruction system, shapes are initialized from a segmented point cloud, i.e. a subset of points is assigned to each shape. An obvious constraint is that these points are located within the shape boundary. The points are projected onto the two-dimensional bitmap of the shape. The point size is automatically computed by the average point density such that neighboring points touch each other (see Fig. 5.9).

Small holes in the initial bitmap are removed by applying a morphological close-operation.



(a) Outside (b) Inside (c) No constraint

Figure 5.10: A depth map provides one of three possible constraints on each shape pixel: (a) pixel should be labeled outside due to visibility of more distant point, (b) pixel should be labeled inside due to coincidence of depth value and shape position, (c) no constraint because the shape is behind another object. The gray line denotes the shape, the blue point denotes the measured point in the depth map, the triangle visualizes the position of the depth camera.

The border is not well-defined by the projected points and will usually be optimized by boundary constraints. Therefore, the bitmap is eroded by a few pixels to relax the point cloud constraint at the boundary. The default value is set to three pixels, but can be adapted by the user. If the assigned point set has a low confidence, e.g. due to large noise, better results can be achieved with a larger erosion. On the other hand, if the final reconstruction should stick close to the assigned point set, a smaller erosion can be chosen. In the final bitmap, all pixels which should be labeled inside the shape have the value one. Thus, the values can be directly used as cost R_{out}.

The cost for labeling pixels inside R_{in} is not clearly defined, because parts of the surface might be missing in the point cloud, e.g. due to occlusion. Nevertheless, it is necessary to define parts as being outside of the shape. Otherwise, the shape will be extended to the whole bounding box as long as no other constraints are defined.

We take the inverse of the initial bitmap and erode it by the same amount of pixels as defined above. This bitmap is used as R_{in}, but it is weighted much lower. In our experiments we use a constant value of 0.1 for labeling these pixels as inside. This value is sufficient for restricting the shape boundary when no other constraints are used, but it doesn't have much influence in case of more confident constraints.

5.3.2 Depth Maps

Multiple depth maps provide the same information as point clouds when they are available together with their camera information. Additionally, it is possible to compute where a shape has been occluded by other geometry. Conversely, it can be determined, where a shape is not valid because it is not visible in any camera. The maximum cost value is used when these areas are labeled as inside.

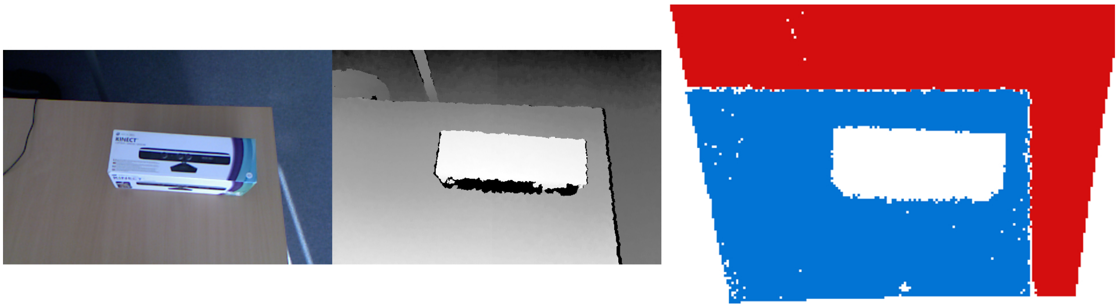


Figure 5.11: Left: Input depth map and color image. Right: Visualization of costs for the plane representing the table top. The blue area is inside the plane surface while the red area is outside. For the white area no information is provided by the depth map.

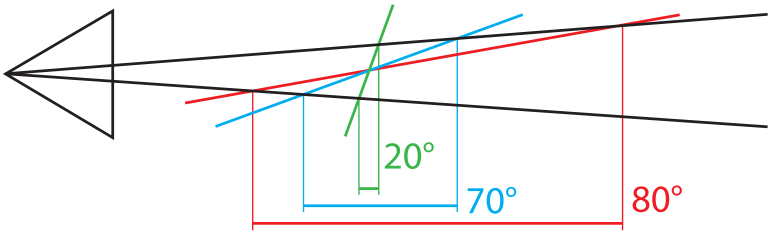


Figure 5.12: For surfaces with normals that form an obtuse angle with the view direction of a range camera, no accurate measurements can be taken from the depth map. The corresponding pixels in the depth map would cover a large area on the surface, as can be compared between the red and green surface. We prune all measurements with normals having a larger angle than 70°.

Each depth map provides three possibilities for each pixel in the shape parameterization, which are illustrated in Fig. 5.10 and 5.11. The pixel has to be outside of the surface, when the corresponding 3D position is inside the depth camera's view frustum, but a more distant point has been captured by the depth camera. R_{in} is set to the maximum value 1 for this pixel. The pixel is inside the surface when the corresponding 3D position approximately coincides with the measured value in the depth map. R_{out} is set to the maximum value 1 for this pixel. No constraint can be created when the shape position is occluded by a closer point in the depth map or when it is not projected onto a depth map pixel with a valid measurement. Both regional terms, R_{in} and R_{out} are set to zero in this case.

When multiple depth maps are available, constraints are created as soon as they occur in one depth map. Care has to be taken for areas that are nearly parallel to the view direction of the depth map. These pixels cover a large area on the shapes and thus do not provide accurate measurements. Therefore we apply a simple pruning step and consider only depth map pixels whose normals form a maximum angle of 70° with the camera direction (see Fig. 5.12). Usually, only a few pixels are affected by these pruning step which can be compensated by more accurate measurements from other range images.

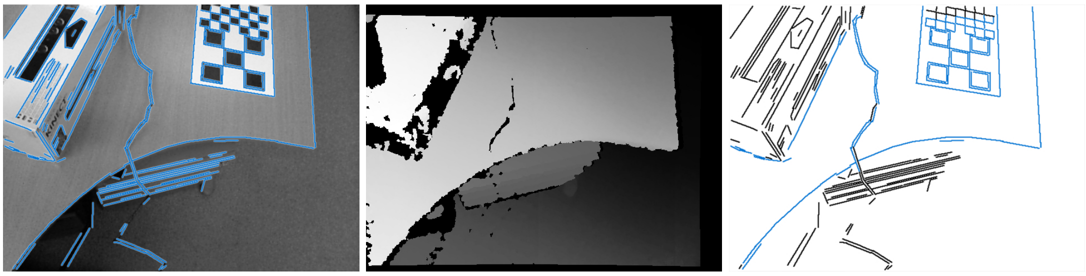

(a) (b) (c)

Figure 5.13: These images illustrate, which line segments are extracted and accepted from one image for a planar shape representing the table top. (a) shows all detected lines in the color image, (b) is the depth image, (c) displays accepted line segments in blue. Line segments that originate from surfaces above or beneath the table top are not used as constraint.

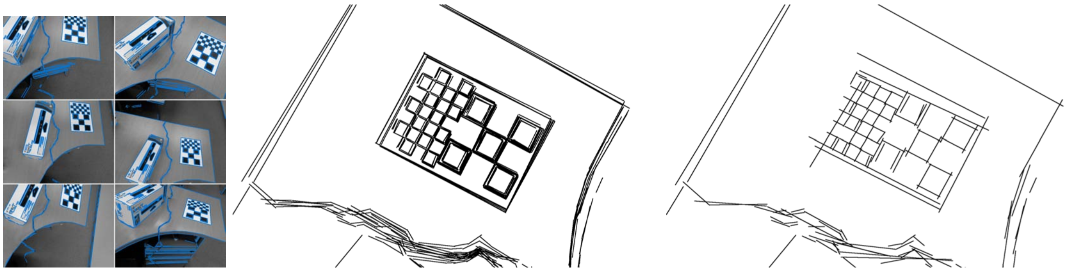

(a) (b) (c)

Figure 5.14: Line segments from six input frames (a) have been projected onto a plane shape (b). They do not overlap exactly due to inaccuracies, thus nearby line segments are merged (c).

5.3.3 Shape Intersections

Intersections with nearby shapes provide very strong cues for surface boundaries. Intersection lines are computed for each pair of shapes and projected into 2D space. The lines have to fall at least in one of the shapes' bounding boxes in order to be accepted. The intersection lines are weighted inversely proportional to the distance between the shapes such that neighboring shapes will be preferred.

The usage of shape intersections is necessary for generating closed models. Additionally, more distant shapes that do not intersect in reality also provide useful information, e.g. when parts of a model are missing in the point cloud.

5.3.4 Images

Color images provide valuable information about the location of surface boundaries. For this it is necessary that an object is clearly distinguishable from its background, e.g. by luminance, color or texture. Unfortunately, such changes often occur inside an object too, which makes it hard to extract edges. Therefore we apply some additional conditions to avoid cluttering from too many image edges.

We use a state-of-the-art line segment detector [49] for detecting line segments in all input color images. The restriction to line segments is useful as we focus on interior scenes and other man-made objects where mainly straight line segments occur. Nevertheless, other shape boundaries are approximated by concatenating multiple lines, provided their visual cue is strong enough (e.g. the table in Fig. 5.13(a)).

Another restriction is based on depth maps registered with the color images, which are available when an RGB-D sensor was used for capturing. For each shape, only line segments which are located on the shape's surface are used as constraint. Line segments where the associated depth values do not meet the shape are removed (see Figure 5.13). Lines along surface boundaries are usually located along large depth steps, but usually they are slightly off due to registration or rasterization errors. Therefore, it is important to look for corresponding depth values in a slightly increased test area. In our system we empirically set this offset to 1.5 pixels which works well for depth maps captured with Microsoft Kinect. The line segments are accepted if any depth measurement inside this offset corresponds to the position of the shape.

Line segments are detected in all available color images and projected onto the 2D parameterization of a shape. Due to inaccuracies in camera pose estimation, shape detection and line extraction, lines are usually projected with small offsets next to each other. Therefore, we merge nearby lines with a line segment clustering algorithm [96], as can be seen in Figure 5.14.

5.3.5 Coplanarity

Coplanarity provides boundary constraints in areas where points from multiple shape boundaries are located approximately on the same plane. This plane provides a joint limit in one direction for multiple shapes. Such limits are valuable for incomplete models, e.g. to restrict walls of an interior room where the ceiling hasn't been scanned. Furthermore, the detected planes are often useful for extending the existing model.

The coplanarity constraint is computed based on the current set of shape boundaries and is updated after each optimization step. We restrict the search to planes perpendicular to global directions of the model, e.g. to plane normals or cylinder axes. This restriction is based on the assumption that shapes in man-made objects are often positioned with orthogonal angles to each other [82]. Otherwise, undesired constraint lines would affect the optimization because planes are detected in unrelated shape boundaries. Sample points are extracted together with their line directions from all current 3D shape boundaries. For each global direction, all sample points with a perpendicular direction are selected. The points are projected to one-dimensional ray coordinates along the global direction. The ray coordinates are clustered in order to find dominant planes.

The clusters are accepted only if the points originate from at least two different shapes and if the points are not approximately collinear. Clusters which are consistent with already existing planes in the model are also removed. The final planes are moved to the median value of all ray coordinates. This ensures that the plane is snapped to dominant boundaries, while averaging over all ray coordinates would be influenced too much by outliers. Figure 5.15 shows an example for optimizing a model with coplanarity constraints.

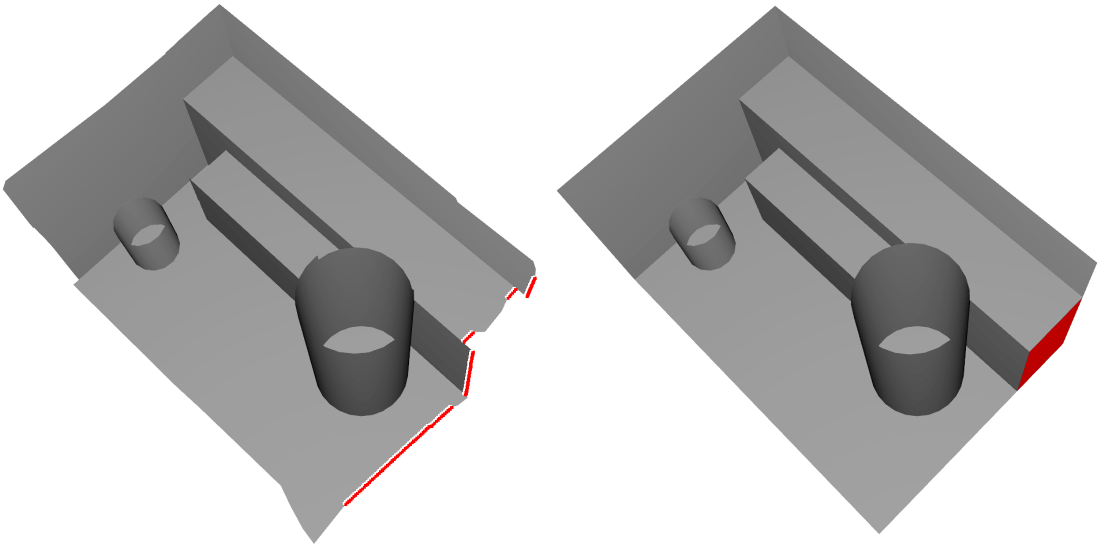


Figure 5.15: Left: Input model with coplanar sample points highlighted in red. Right: Result after optimization with coplanarity constraint. The detected plane has also been added to the model and is shown in red.

5.3.6 2D Shapes

Similar to finding shapes in 3D, local boundaries can be enhanced by approximating them with two-dimensional shapes such as circles or lines. Using two-dimensional shapes generally simplifies the existing polygon. Noisy curves are straightened and small features are removed, while at the same time sharp corners between different shapes are generated. Figure 5.16 shows an example, where a circle and multiple lines are detected in the existing boundary and used as constraint for the next optimization step. As can be seen, the detection of 2D shapes usually needs some iterations of alternating constraint updates and shape optimizations.

The detection of 2D shapes is based on current shape boundaries, as we already have seen with the coplanarity constraints. Points and directions are sampled for a RANSAC-based detection of 2D shapes. Candidate shapes are generated and tested against all sample points. The candidate shape which is supported by the largest number of points is accepted and further shapes are detected in the remaining sample points.

5.3.7 User Input

Fully automatic reconstruction systems not always lead to the desired results. Especially, when large areas of the original scene are missing in the captured data, it is not always possible to reconstruct these missing areas automatically. Therefore it is essential to provide a possibility for manually editing the reconstruction results and including additional information and creativity provided by a human user.

In our reconstruction system, the user can add optimization constraints by explicitly defining

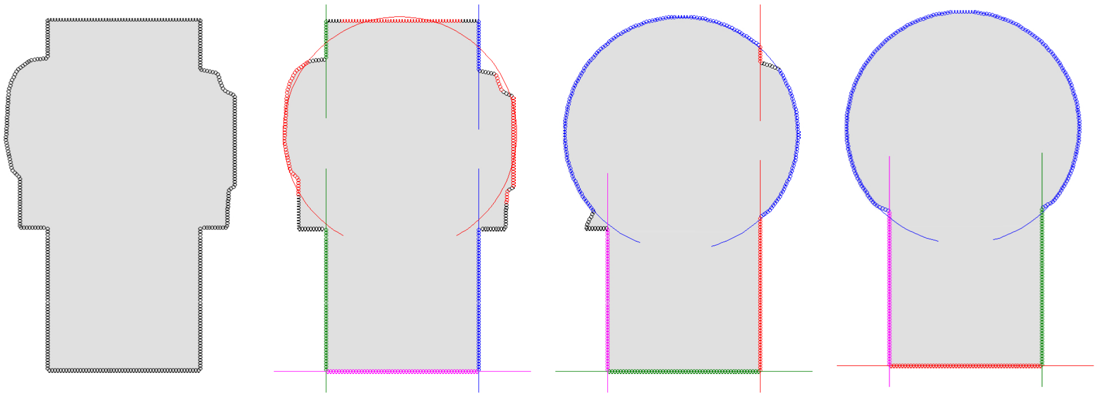


Figure 5.16: 2D shapes are extracted from the shape boundary. The pictures, from left to right, show the results of iteratively updating the constraints and optimizing the shape. In the beginning, parts in the top of the boundary were wrongly attracted by an intersection constraint, but these were removed within three iterations without any user input. Sample points are shown in the same color as their assigned 2D shapes. The colors denote the order of detection (blue, red, green, magenta) in the RANSAC-loop. Black points are not assigned to a 2D shape.

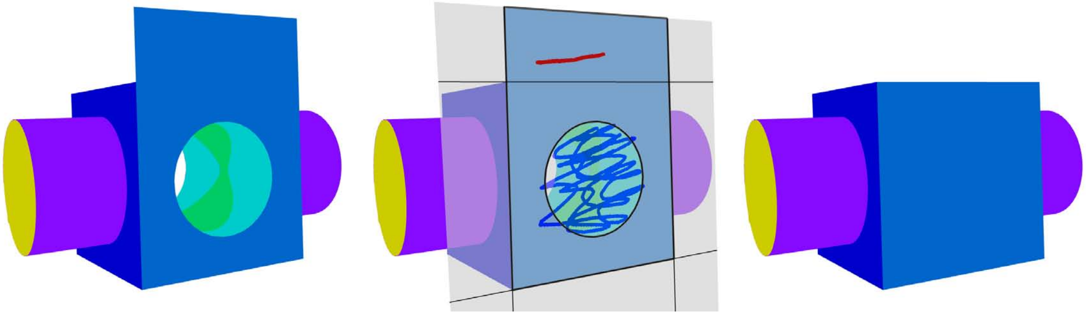


Figure 5.17: User input: Red pixels mark areas as outside, and blue as inside. The right image shows the result after another optimization step.

pixels as *inside* or *outside*. Similarly, it is possible to manually define good boundary positions. These user constraints generally have a very high impact so that they overrule other constraints.

We provide a brushing user interface, where the user can directly draw on the shape in 3D (see Fig. 5.17). This is very easy to use because no knowledge about 3D modeling is needed. In the following optimization step, areas will be filled or deleted according to the new input and the surface boundaries will snap to appropriate boundary constraints.

Some other constraints (coplanarity, 2D shapes) depend on the current shape boundary. A good strategy is therefore to push the shape boundary approximately to the desired location with user input. Afterwards, the constraints are updated and refined to exact borders from other relations.

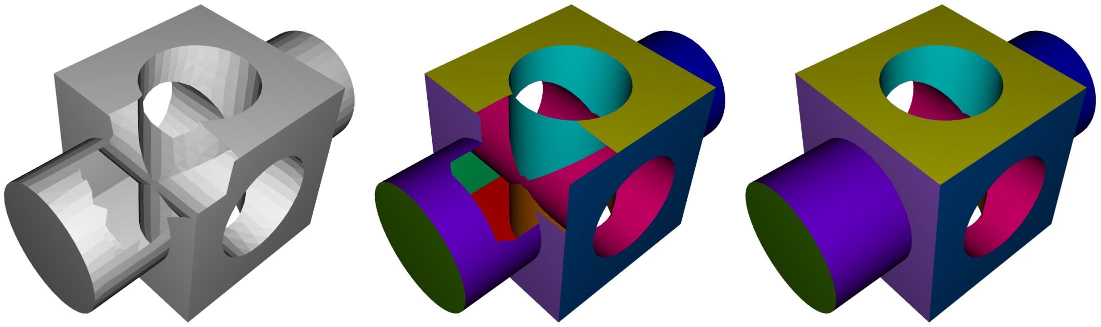
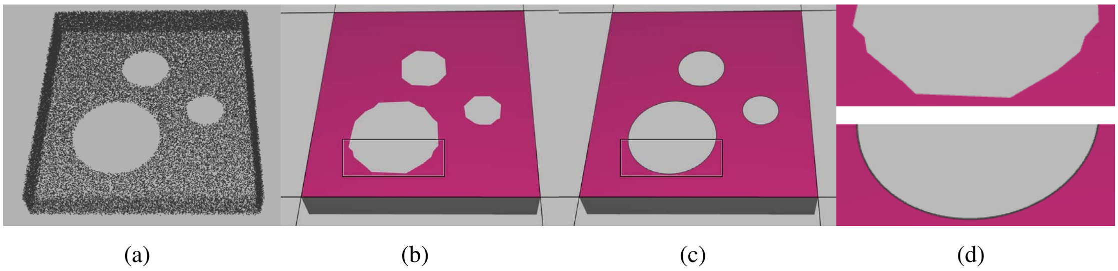


Figure 5.18: Reconstruction of block shape with missing data. Left: original mesh used for sampling. Middle: reconstruction with equal impact factors for point cloud and intersections. Right: high impact factor for intersections.



(a) (b) (c) (d)

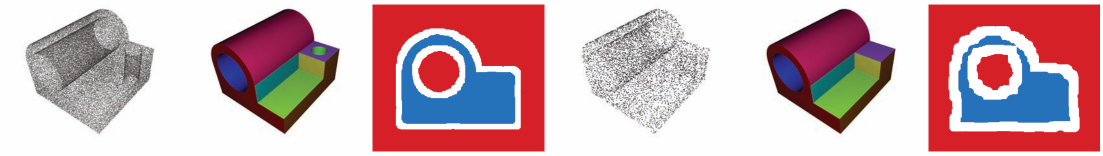
Figure 5.19: A synthetic dataset of a table with three holes has been reconstructed: (a) Sampled point cloud, (b) initial result without 2D shapes constraint, (c) final result based on detected 2D shapes, (d) detailed view of initial and final result.

5.4 Evaluation

We have tested our framework on a variety of synthetic and real-world data sets. The real-world data sets have either been captured as multiple depth maps with Microsoft Kinect (Fig. 5.22, 5.21, and 5.23), created with photogrammetric reconstruction from multiple images (Fig. 5.24), or acquired with a laser scanner (Fig. 5.24).

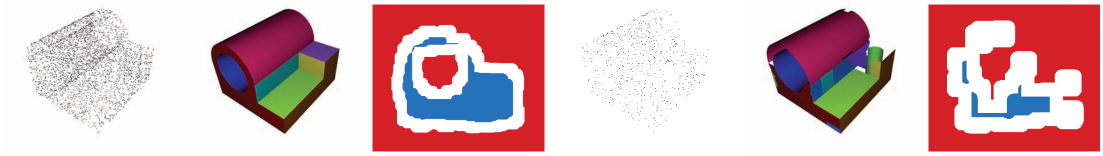
5.4.1 Synthetic datasets

Figure 5.18 shows how our algorithm can deal with missing data. If the boundaries are optimized under equally weighted constraints for point clouds and intersections, the resulting model is approximately consistent with the input mesh. On the other hand, if the impact factor for intersections is largely increased by a factor of 20, the shapes extend to their nearest intersection lines and a closed model is generated. Note, that this solution can only be applied when a closed model is desired. Another solution in this case is user input, where the missing areas can be filled just with a few simple brushing operations.



(a) 69544 points, 12 correct shape boundaries

(b) 6954 points, 10 correct shape boundaries



(c) 3477 points, 8 correct shape boundaries

(d) 695 points, 2 correct shape boundaries

Figure 5.20: Reconstruction of the joint model (consisting of 3 cylinders and 9 planes) with decreasing point density (100%, 10%, 5% and 1%). The images show from left to right: input point set, reconstructed model, point cloud constraint. Note that the shape primitives were not automatically fit in this example, in order to compare only the boundary optimization.

The application of 2D shape constraints has been evaluated as shown in Figure 5.19. A table top with three circular holes has been uniformly sampled. Figure 5.19(b) shows the result when the boundaries are optimized only according to the point cloud and intersections. The holes have been correctly preserved as the surface is densely sampled without large artifacts or missing data. But the surface boundary is not clean because it is just interpolated between the raster cells. After applying the 2D shape constraint, all three circles are correctly detected and the next optimization step produces a surface boundary with exact circles.

Figure 5.20 shows how our algorithm performs with low point densities. The models are reconstructed solely with intersections and the point cloud constraint. The same parameters have been used for all input point sets. As can be seen, good results are obtained also in case of very few points. In the last case, where only one percent of the original point set is used, it is still possible to partially extract correct shape boundaries.

5.4.2 Real-world datasets

Figures 5.21 and 5.22 show the results for two datasets containing a kitchen. The models have been optimized using constraints from the point cloud, intersections and coplanarity as well as user input. In both scenes, large parts are missing in the point cloud. While these parts are still missing in the initially reconstructed point cloud, they can be easily filled in by the user. In the model of Figure 5.22 many additional planes have been inserted based on the coplanarity constraint, e.g. the top and bottom sides of cupboards. In this example it can also be seen, how the coplanarity constraints limits all shapes to the open front side.

Figure 5.23 shows the reconstruction of a bar table. The result of the automatic pipeline, including point cloud, intersection and coplanarity constraints, produced good results for most

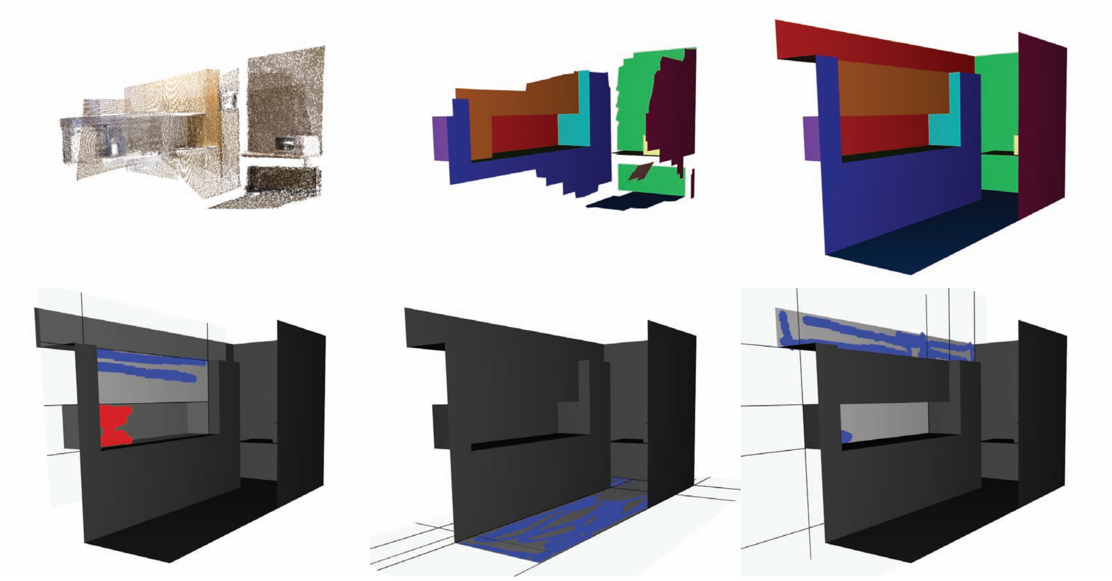


Figure 5.21: This point cloud from eight Kinect range images has been reconstructed with 17 shapes. The top row shows the input point cloud, the initially reconstructed and the final model after some user inputs. The second rows show the user input for some shapes.

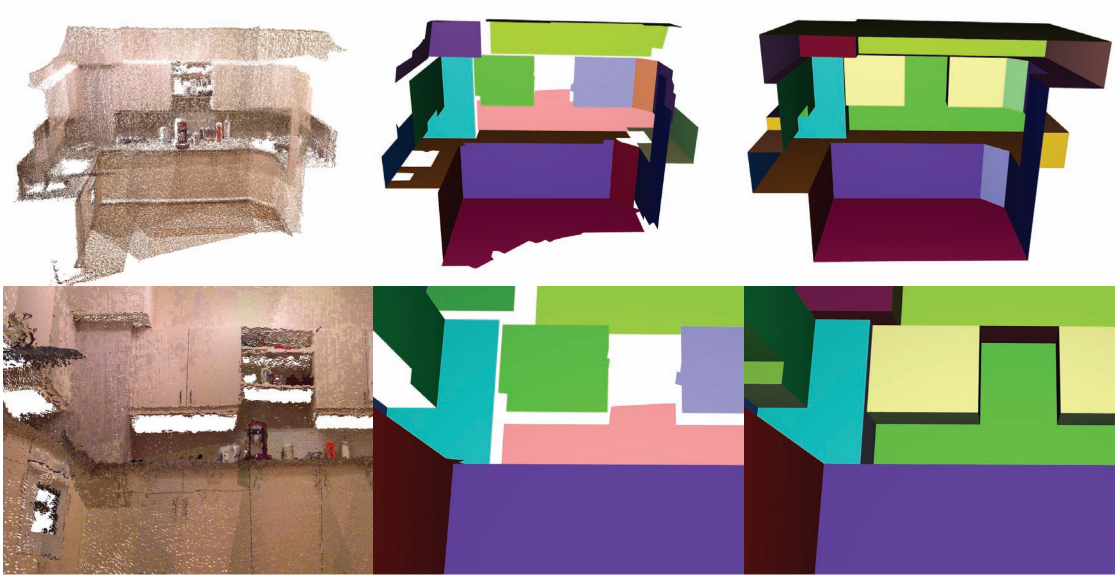


Figure 5.22: A point cloud from thirteen range images has been reconstructed with 26 shapes. Both rows show the input point cloud, the automatically reconstructed and the final model after some user inputs from different viewpoints.

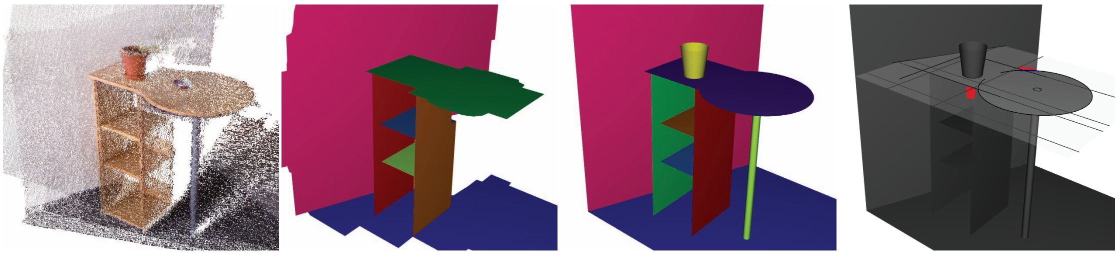


Figure 5.23: Reconstruction of a bar table. The table leg and the flower pot have been manually created by selecting the according points. The right picture shows how the boundary is restricted by the 2D shape constraint.

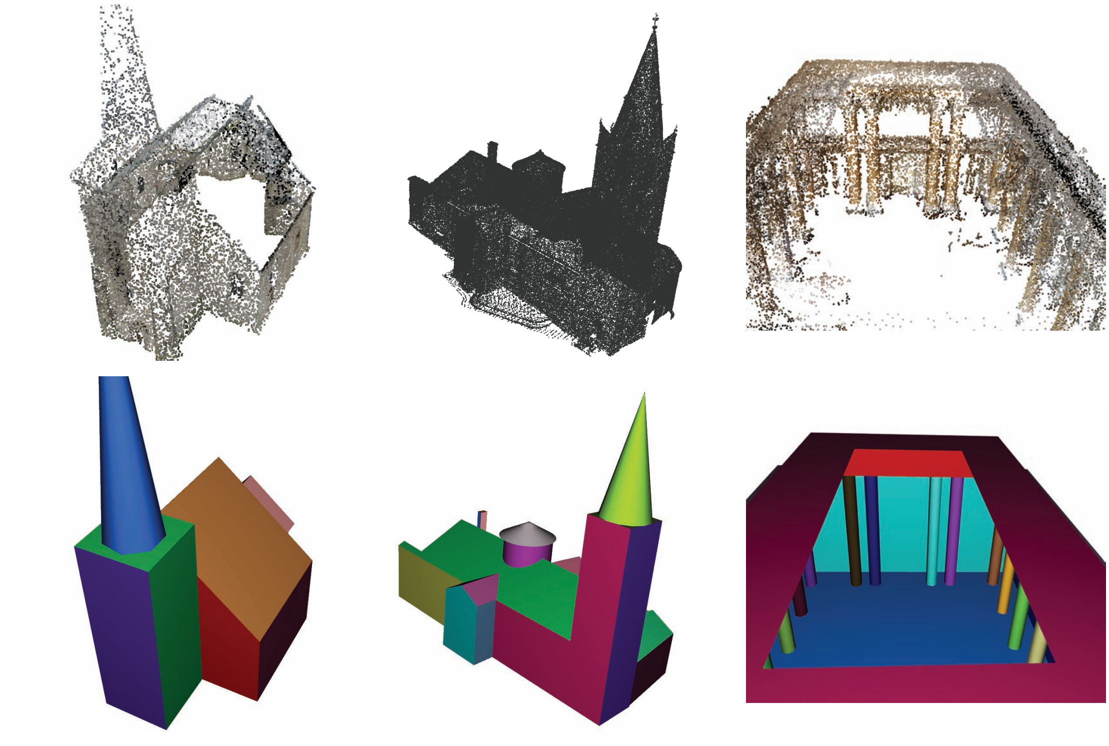


Figure 5.24: Reconstruction of several data sets (from left to right): church (photogrammetry), church of Lans Le Villard (laser scan), ballroom (photogrammetry).

Table 5.2: The table shows the computation times (in seconds or minutes) for shape detection and fitting (T_{pre}), initial automatic boundary optimization (T_{init}), and complete user input (T_{ui}). Additionally, the times for updating constraints (T_{cstr}) and optimizing shapes (T_{opt}) are given. Finally, some statisics on the input and output data are given, namely the number of Kinect RGB-D images (N_{frames}) or the number of input points (N_{points}), as well as the number of shape primitives before and after user input (N_{shapes}), the average number of raster points of the shape primitives (N_{cells}), and the average number of optimization steps per shape primitive (N_{steps}).

	Kitchen 1	Kitchen 2	Bar table	Church	Lans le V.	Ballroom
	Fig. 5.21	Fig. 5.22	Fig. 5.23	Fig. 5.24	Fig. 5.24	Fig. 5.24
T_{pre}	26s	57s	51s	53s	178s	528s
T_{init}	1.3s	7.6s	1.3s	20.1s	27.9s	36.3s
T_{ui}	6.3m	12.7m	3.1m	12.9m	16.0m	26.9m
T_{cstr}	36.2s	105.2s	4.5s	124.6s	230.6s	247.2s
T_{opt}	5.9s	10.8s	1.9s	15.2s	22.3s	50.5s
N_{frames}	8	13	5			
N_{points}				29600	44187	61699
N_{shapes}	12/17	16/26	7/9	18/25	27/32	44/40
N_{cells}	21763	32765	8937	36139	43758	44978
N_{steps}	5.4	6.2	6.1	6.5	5.6	5.5

of the shapes. The wall and the floor needed some manual input in order to generate nice outlines, because the point cloud was only partially visible and there were no limitations by other shapes. For the table top we applied the 2D shape constraint in order to retrieve an exact circular boundary. Details about the 2D shape fitting can be seen in Figure 5.16. The shapes for the table leg and the flower pot weren't discovered by the automatic shape detection algorithm, but our user interface provides the possibility to select points from the input point cloud and fit a shape primitive to it.

Additional examples that have been generated with our reconstruction system can be seen in Figure 5.24. The input data sets are point clouds, either generated by photogrammetric reconstruction [45] or captured with a laser scanner.

As can be seen in Table 5.2, the Ballroom dataset contains a large number of shape primitives, which can still be handled by our system. Because each shape is optimized individually, the optimization step of our algorithm scales linearly with the number of shape primitives provided they have equal resolutions. Currently, our system is limited by the computations needed for constraint updates which are defined over all shape primitives of the scene (e.g. intersections). This clearly limits the number of shape primitives and therefore makes processing more complex scenes tedious. However, reconstructing more complex scenes would be possible if the computation of constraints is restricted to parts of the scene, e.g. by computing intersections only between nearby shape primitives.

5.5 Conclusions

We have presented a novel system for reconstructing 3D models from point clouds and images, especially from RGB-D images. The reconstruction is based on shape primitives such as planes or cylinders, which is useful for man-made scenes such as buildings or interior scenes. The system is divided into two parts: segmentation of RGB-D images and extracting shape boundaries.

In the first part, we introduced a novel method for segmenting multiple depth maps into shape primitives. We organize the data in a sample point graph, where we exploit information from depth images such as connectivity and knowledge about varying noise levels. Our algorithm is memory and time efficient by dividing it into a global optimization of sample points and into individual segmentations of depth images. In the future, the segmentation algorithm can be improved for ordered image sequences by including temporal information.

In the second part, we have presented a novel method for extracting and optimizing reasonable boundaries for shape primitives. The optimization framework supports many input sources and a wide range of high-level constraints. We have shown that the automatic algorithm provides a good initial reconstruction in many cases, despite of noisy or incomplete point clouds. For difficult cases we provide a simple user interface which allows the user to push the shape boundaries towards preferred locations.

While we only employed three types of shape primitives, namely planes, cylinders and cones, the algorithm can easily be extended to other 2D manifolds. The only requirement is that the shape can be parameterized in a two-dimensional grid. An interesting approach would be the usage of NURBS in order to apply the reconstruction algorithm to arbitrary types of scenes. Also the range of constraints can be further extended. For example, constraints could be inferred from approximate symmetries.

CHAPTER 6

Conclusions and Outlook

In this dissertation we presented a set of algorithms for reconstructing architectural scenes and other man-made shapes. The algorithms are based on multi-view imagery or on point sets, and we also combine both input sources. We have focused on methods using geometric primitives in order to generate simple 3D models.

6.1 Conclusions

We presented three novel approaches for reconstructing 3D models of man-made scenes. All three techniques have in common, that simple 3D models are generated, which is achieved by fitting geometric primitives to the input data. Reconstructing simple 3D models has several advantages, e.g. they can be easily edited in post-processing tasks. However, the usage of geometric primitives is not useful for all types of scenes and restricts reconstruction algorithms to man-made scenes. But since a large amount of real-world reconstruction takes place in such environments, it is useful to provide specialized solutions.

The approaches use different input sources, which are either multi-view imagery or 3D point sets. We also show how information from images and point sets can be combined. The algorithms differ in the amount of necessary user input, ranging from an interactive tool (Chapter 3), over optional user input (Chapter 5) to a fully automatic approach (Chapter 4). In summary, the following contributions have been presented:

- For the first algorithm, described in Chapter 3, it is sufficient to have a set of multiple registered images of a scene. We have shown scenes, mainly from industrial environments, for which automatic image matching and thus 3D computations fail due to texture-poor and highly reflective surfaces. For this reason, we proposed a novel interactive technique, where the user approximately outlines edges and contours of a geometric primitive in two or more images. The accurate parameters of the model and its position and orientation are automatically fit to the provided constraints. Additional segmentation and model initialization modules for tubes and pipes makes this algorithm perfectly tailored for reconstructing industrial environments, as can be seen in the provided examples.

- In Chapter 4 we presented an automatic algorithm for reconstructing buildings from point sets. The automatic approach is possible, because dense point sets can be generated from outdoor scenes with photogrammetry or laser scans. The reconstruction is restricted to piecewise-planar models, which is a reasonable approximation for many buildings. Additionally we exploit the observation, that many surfaces of buildings are arranged with right angles or other predefined constellations such as $45°$ or $60°$. Finally, we show that the extraction of boundaries for the reconstructed planes can be improved by analyzing strong edges in input images that are registered to the point cloud.

- Combining information from point clouds and images was further investigated in Chapter 5, where we especially focused on RGB-D images which combine color images with depth images. This algorithm is based on geometric primitives and can be applied to man-made scenes such as interior scenes, CAD models, and buildings as well. We first describe an efficient algorithm for segmenting RGB-D images into multiple geometric primitives. Then we present a novel optimization framework for generating reasonable surface boundaries for the detected geometric primitives. The main advantage of the optimization framework is that multiple input sources can be easily combined. We show how information from point sets, depth maps, and images can be used for detecting good surface boundaries. Additonally, we present how constraints such as shape intersections or coplanarity can be included together with optional user input.

6.2 Outlook

Surface reconstruction is a challenging topic due to the wide variability of input data. The quality of input data highly depends on the scene and surface materials, the acquision technique, and last but not least on the time and carefulness spent by the user to capture a scene on-site. Thus, the data is often noisy, contains outliers, or even large parts of the scene are missing. Generally, there are two solutions for tackling these problems, especially for filling in missing data: Putting the user into the reconstruction loop, or using prior information and extensive knowledge about the type of scene.

We believe, that interactive tools will be important for many applications using reconstruction. Providing important information in advance is often more practical than detecting and resolving possible reconstruction errors in a post-processing step. An important question for future reconstruction techniques will be, which tasks can be efficiently and robustly done by automatic algorithms and which ones can be easily performed by a human operator.

However, interactive applications are not suitable for all tasks of 3D reconstruction. For example, they are hardly scalable to large reconstructions such as whole cities. For autonomous systems such as robots it is of course also not possible to include manual interventions from a human user. The main challenge for these applications will be to automatically choose the correct prior information depending on the contents of a scene.

Extracting high-level constraints from the input data is essential for providing useful interpolations for missing data. As shown in our work, using geometric primitives and global relations provides useful approximations for a wide range of scenes. Depending on the input scene, it might be useful to exploit symmetric arrangements, equal angles or distances, rectangular or other predefined structures, multiple instances of one object, etc. A promising approach is using databases of 3D models for reconstructing scenes [73]. 3D models can be detected when they are only partially visible, and provide perfect extensions for the missing part.

Further combinations of different input sources promise improvements of reconstruction techniques. As we have shown in this thesis, combining point sets with images benefits from the complementary characteristics that these input data provide. Additional input data can be included, such as GPS data or inertial data [109]. For reconstructing interior scenes valuable information can be extracted from floor plans. Similarly, reconstructing cities and buildings can be enhanced with information from city maps. Modern sensing products, such as mobile phones or depth cameras, provide inertial data measured with accelerometers and gyroscopes, which can be used for faster and more accurate acquisition. The equipment of mobile phones might be extended in the future, e.g. by stereo cameras or other depth sensors. This will open up the field of 3D reconstruction to many customers and lead to new applications.

Computer vision algorithms are suitable for mobile phones and other mobile devices since these provide cameras and powerful processors. The main challenge for these devices are the low memory and performance capabilities which require sophisticated algorithms. Furthermore, applications have to deal with uncalibrated input data and a wide range of different hardware and software components. On the other hand, mobile devices provide an easy way for distributing applications to many users. For example, augmented reality provides interesting applications for mobile devices using 3D reconstruction. Current approaches on mobile devices depend on visual markers and predefined image targets. For an immersive experience it is necessary that virtual and real objects interact with each other. Automatic reconstruction techniques that provide 3D models in real-time are crucial for enabling such interactions in unknown environments. Dense multi-view reconstruction algorithms [99] already provide promising results, but currently they are not yet supported on standard devices. We believe, that in the future surface reconstruction, object recognition, and analyzing structures of a scene will be important topics for augmented reality.

In conclusion, 3D reconstruction is still a very challenging topic that is far from being solved. We believe, that different types of algorithms are necessary in order to handle the wide range of input data as well as many different characteristics of desired output data.

Bibliography

[1] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Brian Curless, Steven M Seitz, and Richard Szeliski. Reconstructing Rome. *IEEE Computer*, pages 40–47, 2010.

[2] Sameer Agarwal, Noah Snavely, Ian Simon, Steven M Seitz, and Richard Szeliski. Building Rome in a Day. *IEEE International Conference on Computer Vision (ICCV)*, pages 72–79, September 2009.

[3] Pierre Alliez, David Cohen-Steiner, Yiying Tong, and Mathieu Desbrun. Voronoi-based variational reconstruction of unoriented point sets. In *Eurographics Symposium on Geometry Processing*, pages 39–48, 2007.

[4] Murat Arikan, Michael Schwärzler, Simon Flöry, Michael Wimmer, and Stefan Maierhofer. O-Snap: Optimization-Based Snapping for Modeling Architecture. *ACM Transactions on Graphics (TOG)*, 2013.

[5] Autodesk. Image Modeler.

[6] Haim Avron, Andrei Sharf, Chen Greif, and Daniel Cohen-Or. l1-sparse Reconstruction of Sharp Point Set Surfaces. *ACM Transactions on Graphics (TOG)*, 29(5):135, 2010.

[7] Caroline Baillard and Andrew Zisserman. Automatic reconstruction of piecewise planar models from multiple views. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 559–565. IEEE Comput. Soc, 1999.

[8] Atsuhiko Banno, Tomohito Masuda, Takeshi Oishi, and Katsushi Ikeuchi. Flying Laser Range Sensor for Large-Scale Site-Modeling and Its Applications in Bayon Digital Archival Project. *International Journal of Computer Vision*, 78(2-3):207–222, 2007.

[9] Alan H Barr. Global and local deformations of solid primitives. In *SIGGRAPH*, pages 21–30, 1984.

[10] Adrien Bartoli. A random sampling strategy for piecewise planar scene segmentation. *Computer Vision and Image Understanding*, 105(1):42–59, 2007.

[11] Alexander Berner, Michael Wand, Niloy J Mitra, Daniel Mewes, and Hans-Peter Seidel. Shape Analysis with Subspace Symmetries. *Computer Graphics Forum*, 30(2):277–286, 2011.

[12] Jan Böhm. Facade Detail From Incomplete Range Data. *The International Archives of the Photogrammetry, Remove Sensing and Spatial Information Sciences*, XXXVII:653–658, 2008.

[13] Jan Böhm and Norbert Haala. Efficient integration of aerial and terrestrial laser data for virtual city modeling using lasermaps. In *ISPRS workshop laser scanning*, pages 192–197, 2005.

[14] Martin Bokeloh, Alexander Berner, Michael Wand, Hans-Peter Seidel, and Andreas Schilling. Symmetry Detection Using Feature Lines. *Computer Graphics Forum*, 28(2):697–706, 2009.

[15] Yuri Boykov and Marie-Pierre Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in ND images. In *IEEE International Conference on Computer Vision (ICCV)*, pages 105–112, 2001.

[16] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast Approximate Energy Minimization via Graph Cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[17] Claus Brenner. Building reconstruction from images and laser scanning. *International Journal of Applied Earth Observation and Geoinformation*, 6(3-4):187–198, 2005.

[18] James B Campbell and Randolph H Wynne. *Introduction to Remote Sensing*. Guilford Publications, 5th edition, 2011.

[19] Jonathan C Carr, Richard K Beatson, Jon B Cherrie, Toby J Mitchell, W Richard Fright, Bruce C McCallum, and Tim R Evans. Reconstruction and representation of 3D objects with radial basis functions. In *SIGGRAPH*, pages 67–76, New York, New York, USA, August 2001. ACM Press.

[20] Duygu Ceylan, Niloy J Mitra, Hao Li, Thibaut Weise, and Mark Pauly. Factored Facade Acquisition using Symmetric Line Arrangements. *Computer Graphics Forum*, 31(2):671–680, May 2012.

[21] Will Chang and Matthias Zwicker. Global registration of dynamic range scans for articulated model reconstruction. *ACM Transactions on Graphics (TOG)*, 30(3):1–15, May 2011.

[22] Anne-Laure Chauve, Patrick Labatut, and Jean-Philippe Pons. Robust piecewise-planar 3D reconstruction and completion from large-scale unstructured point data. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1261–1268, 2010.

[23] Jie Chen and Baoquan Chen. Architectural Modeling from Sparsely Scanned Range Data. *International Journal of Computer Vision*, 78(2-3):223–236, 2008.

[24] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799, 1995.

[25] Filiberto Chiabrando, Roberto Chiabrando, Dario Piatti, and Fulvio Rinaudo. Sensors for 3D Imaging: Metric Evaluation and Calibration of a CCD/CMOS Time-of-Flight Camera. *Sensors*, 9(12):10080–96, 2009.

[26] Roberto Cipolla and Duncan Robertson. 3D models of architectural scenes from uncalibrated images and vanishing points. In *International Conference on Image Analysis and Processing (ICIAP)*, pages 824–829, 1999.

[27] Andrea Cohen, Christopher Zach, Sudipta N Sinha, and Marc Pollefeys. Discovering and exploiting 3D symmetries in structure from motion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1514–1521, 2012.

[28] Hugo Cornelius and Gareth Loy. Detecting Bilateral Symmetry in Perspective. In *Computer Vision and Pattern Recognition Workshop (CVPRW)*, page 191, 2006.

[29] Antonio Criminisi, Ian Reid, and Andrew Zisserman. Single View Metrology. *International Journal of Computer Vision*, 40(2):123–148, 2000.

[30] Brian Curless and Marc Levoy. A Volumetric Method for Building Complex Models from Range Images Volumetric integration. In *SIGGRAPH*, pages 303–312, 1996.

[31] Paul E Debevec, Camillo J Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In *SIGGRAPH*, pages 11–20, 1996.

[32] Anthony Dick, Philip H S Torr, and Roberto Cipolla. Modelling and Interpretation of Architecture from Several Images. *International Journal of Computer Vision*, 60(2):111–134, 2004.

[33] Hao Du, Peter Henry, Xiaofeng Ren, Marvin Cheng, Dan B Goldman, Steven M Seitz, and Dieter Fox. Interactive 3D Modeling of Indoor Environments with a Consumer Depth Camera. In *International Conference on Ubiquitous Computing (UbiComp)*, pages 75–84, 2011.

[34] Sabry El-Hakim, Emily Whiting, and Lorenzo Gonzo. 3D Modeling with Reusable and Integrated Building Blocks. In *Conference on Optial 3-D Measurement Techniques*, 2005.

[35] Martin A Fischler and Robert B Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6), 1981.

[36] Matthew Fisher and Pat Hanrahan. Context-based search for 3D models. *ACM Transactions on Graphics (TOG)*, 29(6):182, 2010.

[37] Shachar Fleishman, Daniel Cohen-Or, and Claudio T Silva. Robust moving least-squares fitting with sharp features. *ACM Transactions on Graphics (TOG)*, 24(3):544–552, 2005.

[38] Jan-Michael Frahm, Pierre Fite-Georgel, David Gallup, Tim Johnson, Rahul Raguram, Changchang Wu, Yi-Hung Jen, Enrique Dunn, Brian Clipp, Svetlana Lazebnik, and Marc Pollefeys. Building Rome on a Coudless Day. *European Conference on Computer Vision (ECCV)*, pages 368–381, 2010.

[39] Friedrich Fraundorfer, Konrad Schindler, and Horst Bischof. Piecewise planar scene reconstruction from sparse correspondences. *Image and Vision Computing*, 24(4):395–406, 2006.

[40] Christian Frueh and Avideh Zakhor. 3D model generation for cities using aerial photographs and ground level laser scans. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 31–38, 2001.

[41] Yasutaka Furukawa, Brian Curless, Steven M Seitz, and Richard Szeliski. Manhattan-world stereo. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1422–1429, 2009.

[42] Yasutaka Furukawa, Brian Curless, Steven M Seitz, and Richard Szeliski. Reconstructing building interiors from images. In *IEEE International Conference on Computer Vision (ICCV)*, pages 80–87, 2009.

[43] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[44] Yasutaka Furukawa and Jean Ponce. Accurate camera calibration from multi-view stereo and bundle adjustment. *International Journal of Computer Vision*, 84(3):257–268, 2009.

[45] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1362–1376, 2010.

[46] Ran Gal, Ariel Shamir, Tal Hassner, Mark Pauly, and Daniel Cohen-Or. Surface reconstruction using local shape priors. In *Eurographics Symposium on Geometry Processing*, pages 253–262, 2007.

[47] Michael Goesele, Brian Curless, and Steven M Seitz. Multi-View Stereo Revisited. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2402–2409, 2006.

[48] Giorgio Grisetti, Slawomir Grzonka, Cyrill Stachniss, Patrick Pfaff, and Wolfram Burgard. Efficient Estimation of Accurate Maximum Likelihood Maps in 3D. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3472–3478, 2007.

[49] Rafael Grompone von Gioi, Jérémie Jakubowicz, Jean-Michel Morel, and Gregory Randall. LSD: a fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4):722–732, April 2010.

[50] Gabriele Guidi, Laura Micoli, Michele Russo, Monica De Simone, Alessandro Spinetti, and Luca Carosso. 3D digitization of a large model of imperial Rome Bernard Frischer. In *International Conference on 3-D Digital Imaging and Modeling (3DIM)*, pages 565–572, 2005.

[51] Norbert Haala, Susanne Becker, and Martin Kada. Cell decomposition for the generation of building models at multiple scales. In *IAPRS Symposium Photogrammetric Computer Vision*, pages 19–24, 2006.

[52] Norbert Haala and Martin Kada. An update on automatic 3D building reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing*, 65(6):570–580, 2010.

[53] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[54] Daniel Cabrini Hauagge and Noah Snavely. Image matching using local symmetry features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 206–213, 2012.

[55] Varsha Hedau, Derek Hoiem, and David Forsyth. Recovering the spatial layout of cluttered rooms. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1849–1856, 2009.

[56] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In *International Symposium on Experimental Robotics (ISER)*, 2010.

[57] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments. *The International Journal of Robotics Research*, 31(5):647–663, February 2012.

[58] Vu Hoang Hiep, Renaud Keriven, Patrick Labatut, and Jean-Philippe Pons. Towards high-resolution large-scale multi-view stereo. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1430–1437, 2009.

[59] Michael Hornacek and Stefan Maierhofer. Extracting vanishing points across multiple views. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 953–960, 2011.

[60] Hui Huang, Dan Li, Hao Zhang, Uri Ascher, and Daniel Cohen-Or. Consolidation of unorganized point clouds for surface reconstruction. *ACM Transactions on Graphics (TOG)*, 28(5):176, 2009.

[61] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera. In *ACM Symposium on User Interface Software and Technology*, pages 558–568, 2011.

[62] Aleš Jaklič, Aleš Leonardis, and Franc Solina. *Segmentation and recovery of superquadrics: computational imaging and vision*. Kluwer Academic Publishers, 2000.

[63] Philipp Jenke, Bastian Krückeberg, and Wolfgang Straß er. Surface reconstruction from fitted shape primitives. In *Vision, Modeling, and Visualization (VMV)*, pages 31–40, 2008.

[64] Philipp Jenke, Michael Wand, and Wolfgang Straß er. Patch-Graph Reconstruction for Piecewise Smooth Surfaces. In *Vision, Modeling, and Visualization (VMV)*, pages 3–12, 2008.

[65] Nianjuan Jiang, Ping Tan, and Loong-Fah Cheong. Symmetric architecture modeling with a single image. *ACM Transactions on Graphics (TOG)*, 28(5):113, 2009.

[66] Timor Kadir and Michael Brady. Saliency, scale and image description. *International Journal of Computer Vision*, 45(2):83–105, 2001.

[67] Timo Kahlmann and Hilmar Ingensand. Calibration and development for increased accuracy of 3D range imaging cameras. *Journal of Applied Geodesy*, 2(1):1–11, January 2008.

[68] Wilfried Karel, Sajid Ghuffar, and Norbert Pfeifer. Modelling and Compensating Internal Light Scattering in Time of Flight Range Cameras. *The Photogrammetric Record*, 27(138):155–174, June 2012.

[69] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Eurographics Symposium on Geometry Processing*, pages 61–70, 2006.

[70] Kourosh Khoshelham. Accuracy analysis of kinect depth data. In *ISPRS workshop laser scanning*, 2011.

[71] Kourosh Khoshelham and Sander Oude Elberink. Accuracy and resolution of Kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–54, January 2012.

[72] Vladimir G Kim, Wilmot Li, Niloy J Mitra, Stephen DiVerdi, and Thomas Funkhouser. Exploring Collections of 3D Models using Fuzzy Correspondences. *ACM Transactions on Graphics (TOG)*, 31(4):54, 2012.

[73] Young Min Kim, Niloy J Mitra, Dong-Ming Yan, and Leonidas J Guibas. Acquiring 3D Indoor Environments with Variability and Repetition. *ACM Transactions on Graphics (TOG)*, 31(6):11, 2012.

[74] Vladimir Kolmogorov and Ramin Zabih. Multi-camera Scene Reconstruction via Graph Cuts. In *European Conference on Computer Vision (ECCV)*, pages 82–96, 2002.

[75] Adarsh Kowdle, Yao-Jen Chang, Andrew Gallagher, and Tsuhan Chen. Active learning for piecewise planar 3D reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 929–936, 2011.

[76] Klaus-Dieter Kuhnert and Martin Stommel. Fusion of Stereo-Camera and PMD-Camera Data for Real-Time Suited Precise 3D Environment Reconstruction. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4780–4785, 2006.

[77] Patrick Labatut, Jean-Philippe Pons, and Renaud Keriven. Hierarchical shape-based surface reconstruction for dense multi-view stereo. In *IEEE International Conference on Computer Vision Workshops*, pages 1598–1605, 2009.

[78] Florent Lafarge and Clément Mallet. Creating Large-Scale City Models from 3D-Point Clouds: A Robust Approach with Hybrid Representation. *International Journal of Computer Vision*, 99(1):69–85, 2012.

[79] Franz Leberl, Arnold Irschara, Thomas Pock, Philipp Meixner, Michael Gruber, Susanne Scholz, and Alexander Wiechert. Point Clouds: Lidar versus 3D Vision. *Photogrammetric Engineering and Remote Sensing*, 76(10):1123–1134, 2010.

[80] David Levin. Mesh-independent surface interpolation. In *Geometric modeling for scientific visualization*, pages 1–17, 2003.

[81] Marc Levoy, Jeremy Ginsberg, Jonathan Shade, Duane Fulk, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, and James Davis. The Digital Michelangelo Project: 3D Scanning of Large Statues. In *SIGGRAPH*, pages 131–144, 2000.

[82] Yangyan Li, Xiaokun Wu, Yiorgos Chrysathou, Andrei Sharf, Daniel Cohen-Or, and Niloy J Mitra. Globfit: Consistently fitting primitives by discovering global relations. *ACM Transactions on Graphics (TOG)*, 30(4):52, 2011.

[83] Yangyan Li, Qian Zheng, Andrei Sharf, Daniel Cohen-Or, Baoquan Chen, and Niloy J Mitra. 2D-3D fusion for layer decomposition of urban facades. In *IEEE International Conference on Computer Vision (ICCV)*, pages 882–889, 2011.

[84] David Liebowitz, Antonio Criminisi, and Andrew Zisserman. Creating Architectural Models from Images. *Computer Graphics Forum*, 18(3):39–50, 1999.

[85] Marvin Lindner, Ingo Schiller, Andreas Kolb, and Reinhard Koch. Time-of-Flight sensor calibration for accurate range sensing. *Computer Vision and Image Understanding*, 114(12):1318–1328, December 2010.

[86] Yaron Lipman, Daniel Cohen-Or, David Levin, and Hillel Tal-Ezer. Parameterization-free projection for geometry reconstruction. *ACM Transactions on Graphics (TOG)*, 26(3):22, July 2007.

[87] Manolis I A Lourakis and Antonis A Argyros. The Design and Implementation of a Generic Sparse Bundle Adjustment Software Package Based on the Levenberg-Marquardt Algorithm, 2004.

[88] David G Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[89] Gareth Loy and Jan-Olof Eklundh. Detecting symmetry and symmetric constellations of features. In *European Conference on Computer Vision (ECCV)*, pages 508–521, 2006.

[90] Jiri Matas, O Chum, M Urban, and Tomas Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. In *British Machine Vision Conference (BMVC)*, pages 384–393, 2004.

[91] Niloy J Mitra, Leonidas J Guibas, and Mark Pauly. Partial and approximate symmetry detection for 3D geometry. *ACM Transactions on Graphics (TOG)*, 25(3):560–568, 2006.

[92] Eric N Mortensen and William A Barrett. Interactive Segmentation with Intelligent Scissors. *Graphical Models and Image Processing*, 60(5):349–384, 1998.

[93] Pascal Müller, Gang Zeng, Peter Wonka, and Luc Van Gool. Image-based procedural modeling of facades. *ACM Transactions on Graphics (TOG)*, 26(3):85, 2007.

[94] Przemyslaw Musialski, Christian Luksch, Michael Schwärzler, Matthias Buchetics, Stefan Maierhofer, and Werner Purgathofer. Interactive Multi-View Façade Image Editing. In *Vision, Modeling, and Visualization (VMV)*, pages 131–138, 2010.

[95] Przemyslaw Musialski, Meinrad Recheis, Stefan Maierhofer, Peter Wonka, and Werner Purgathofer. Tiling of ortho-rectified facade images. In *Spring Conference on Computer Graphics (SCCG)*, pages 117–126, 2010.

[96] Peter F M Nacken. A metric for line segments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(12):1312–1318, 1993.

[97] Liangliang Nan, Andrei Sharf, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. Smartboxes for interactive urban reconstruction. *ACM Transactions on Graphics (TOG)*, 29(4):93, 2010.

[98] Liangliang Nan, Ke Xie, and Andrei Sharf. A Search-Classify Approach for Cluttered Indoor Scene Understanding. *ACM Transactions on Graphics (TOG)*, 31(6), 2012.

[99] Richard A Newcombe and Andrew J Davison. Live dense reconstruction with a single moving camera. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1498–1505. Ieee, June 2010.

[100] Richard A Newcombe, Andrew J Davison, Shahram Izadi, Pushmeet Kohli, Otmar Hilliges, Jamie Shotton, David Molyneaux, Steve Hodges, David Kim, and Andrew Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 127–136. Ieee, October 2011.

[101] David Nistér. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–77, June 2004.

[102] David Nistér and Henrik Stewénius. Scalable recognition with a vocabulary tree. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2162–2168, 2006.

[103] Byong Mok Oh, Max Chen, Julie Dorsey, and Frédo Durand. Image-based modeling and photo editing. In *SIGGRAPH*, pages 433–442, 2001.

[104] Mark Pauly, Niloy J Mitra, J Giesen, Leonidas J Guibas, and Markus Gross. Example-based 3D scan completion. In *Eurographics Symposium on Geometry Processing*, pages 23–32, 2005.

[105] Mark Pauly, Niloy J Mitra, Johannes Wallner, Helmut Pottmann, and Leonidas J Guibas. Discovering Structural Regularity in 3D Geometry. *ACM Transactions on Graphics (TOG)*, 27(3):43, 2008.

[106] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. *ACM Transactions on Graphics (TOG)*, 22(3):313–318, 2003.

[107] Norbert Pfeifer and Christian Briese. Geometrical aspects of airborne laser scanning and terrestrial laser scanning. *International Archives of Photogrametry, Remote Sensing and Spatial Information Sciences*, XXXVI(3):311–319, 2007.

[108] Norbert Pfeifer, Camillo Ressl, and Wilfried Karel. Range calibration for terrestrial laser scanners and range cameras. *Proceedings of SPIE*, 7447:1–15, 2009.

[109] Marc Pollefeys, David Nistér, Jan-Michael Frahm, Amir Akbarzadeh, Philippos Mordohai, Brian Clipp, C Engels, David Gallup, S-J Kim, Paul Merrell, C Salmi, Sudipta N Sinha, B Talton, L Wang, Q Yang, Henrik Stewénius, R Yang, G Welch, and H Towles. Detailed Real-Time Urban 3D Reconstruction from Video. *International Journal of Computer Vision*, 78(2-3):143–167, 2007.

[110] Jean-Philippe Pons, Renaud Keriven, and Olivier Faugeras. Modelling Dynamic Scenes by Registering Multi-View Image Sequences. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 822–827, 2005.

[111] Pierre Poulin, Mathieu Ouimet, and Marie-Claude Frasson. Interactively modeling with photogrammetry. In *Eurographics Workshop on Rendering*, pages 93–104, 1998.

[112] Irene Reisner-Kollmann, Anton L. Fuhrmann, and Werner Purgathofer. Interactive reconstruction of industrial sites using parametric models. In *Spring Conference on Computer Graphics (SCCG)*, pages 101–108, 2010.

[113] Irene Reisner-Kollmann, Christian Luksch, and Michael Schwärzler. Reconstructing Buildings as Textured Low Poly Meshes from Point Clouds and Images. In *Eurographics Short Papers*, pages 17–20, 2011.

[114] Irene Reisner-Kollmann and Stefan Maierhofer. Consolidation of Multiple Depth Maps. In *Computer Vision Workshops (ICCV Workshops)*, pages 1120–1126, 2011.

[115] Irene Reisner-Kollmann and Stefan Maierhofer. Segmenting multiple range images with primitive shapes. In *Systems, Signals and Image Processing (IWSSIP)*, pages 320–323, 2012.

[116] Irene Reisner-Kollmann, Stefan Maierhofer, and Werner Purgathofer. Reconstructing shape boundaries with multimodal constraints. *Computers & Graphics*, to appear, 2013.

[117] Xiaofeng Ren, Liefeng Bo, and Dieter Fox. RGB-(D) scene labeling: Features and algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2759–2766, 2012.

[118] Duncan Robertson and Roberto Cipolla. An interactive system for constraint-based modelling. In *British Machine Vision Conference (BMVC)*, pages 536–545, 2000.

[119] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the ICP algorithm. In *International Conference on 3-D Digital Imaging and Modeling (3DIM)*, pages 145–152, 2001.

[120] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). *IEEE International Conference on Robotics and Automation*, pages 1–4, May 2011.

[121] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Mihai Dolha, and Michael Beetz. Towards 3D Point cloud based object maps for household environments. *Robotics and Autonomous Systems*, 56(11):927–941, November 2008.

[122] Frederik Schaffalitzky and Andrew Zisserman. Multi-view matching for unordered image sets, or "How do I organize my holiday snaps?". In *European Conference on Computer Vision (ECCV)*, pages 414–431, 2002.

[123] Daniel Scharstein, Richard Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1):7–42, 2002.

[124] Konrad Schindler and Joachim Bauer. A model-based method for building reconstruction. In *IEEE International Workshop on Higher-Level Knowledge in 3D Modeling and Motion Analysis*, pages 74–82, 2003.

[125] Cordelia Schmid and Andrew Zisserman. The geometry and matching of lines and curves over multiple views. *International Journal of Computer Vision*, 40(3):199–233, 2000.

[126] Ruwen Schnabel, Patrick Degener, and Reinhard Klein. Completion and Reconstruction with Primitive Shapes. *Computer Graphics Forum*, 28(2):503–512, April 2009.

[127] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient RANSAC for Point-Cloud Shape Detection. *Computer Graphics Forum*, 26(2):214–226, June 2007.

[128] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 519–528, 2006.

[129] Tianjia Shao, Weiwei Xu, Kun Zhou, Jingdong Wang, Dongping Li, and Baining Guo. An Interactive Approach to Semantic Modeling of Indoor Scenes with an RGBD Camera. *ACM Transactions on Graphics (TOG)*, 31(5), 2012.

[130] Loic Simon, Olivier Teboul, Panagiotis Koutsourakis, Luc Van Gool, and Nikos Paragios. Parameter-free/Pareto-driven procedural 3D reconstruction of buildings from ground-level sequences. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 518–525, 2012.

[131] Sudipta N Sinha, Krishnan Ramnath, and Richard Szeliski. Detecting and Reconstructing 3D Mirror Symmetric Objects. In *European Conference on Computer Vision (ECCV)*, pages 586–600, 2012.

[132] Sudipta N Sinha, Drew Steedly, and Richard Szeliski. Piecewise planar stereo for image-based rendering. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1881–1888, 2009.

[133] Sudipta N Sinha, Drew Steedly, Richard Szeliski, Maneesh Agrawala, and Marc Pollefeys. Interactive 3D architectural modeling from unordered photo collections. *ACM Transactions on Graphics (TOG)*, 27(5):159, 2008.

[134] Jan Smisek, Michal Jancosek, and Tomas Pajdla. 3D with Kinect. In *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1154–1160. IEEE, November 2011.

[135] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3D. *ACM Transactions on Graphics (TOG)*, 25(3):835–846, 2006.

[136] Noah Snavely, Steven M Seitz, and Richard Szeliski. Skeletal graphs for efficient structure from motion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[137] Noah Snavely, Ian Simon, Michael Goesele, Richard Szeliski, and Steven M Seitz. Scene reconstruction and visualization from community photo collections. *Proceedings of the IEEE*, 98(8):1370–1390, 2010.

[138] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image Processing, Analysis, and Machine Vision*. Thomson-Engineering, 2008.

[139] Todor Stoyanov, Athanasia Louloudi, Henrik Andreasson, and Achim J Lilienthal. Comparative evaluation of range sensor accuracy in indoor environments. In *European Conference on Mobile Robots (ECMR)*, pages 19–24, 2011.

[140] Christoph Strecha, Wolfgang von Hansen, Luc Van Gool, Pascal Fua, and Ulrich Thoennessen. On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[141] Eos Systems. PhotoModeler.

[142] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2011.

[143] Olivier Teboul, Loic Simon, Panagiotis Koutsourakis, and Nikos Paragios. Segmentation of building facades using procedural shape priors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3105–3112, 2010.

[144] Anton van den Hengel, Anthony Dick, Thorsten Thormählen, Philip H S Torr, and Benjamin Ward. Building Models of Regular Scenes from Structure and Motion. In *British Machine Vision Conference (BMVC)*, volume 1, pages 21.1–21.10. British Machine Vision Association, 2006.

[145] Anton van den Hengel, Anthony Dick, Thorsten Thormählen, Philip H S Torr, and Benjamin Ward. Fitting multiple models to multiple images with minimal user interaction. In *International Workshop on the Representation and use of Prior Knowledge in Vision (WRUPKV), in conjunction with ECCV*, volume 1, 2006.

[146] Anton van den Hengel, Anthony Dick, Thorsten Thormählen, and Benjamin Ward. Interactive 3D Model Completion. In *Digital Image Computing Techniques and Applications (DICTA)*, pages 175–181, 2007.

[147] Anton van den Hengel, Anthony Dick, Thorsten Thormählen, Benjamin Ward, and Philip H S Torr. A shape hierarchy for 3D modelling from video. In *GRAPHITE*, volume 1, pages 63–70, 2007.

[148] Anton van den Hengel, Anthony Dick, Thorsten Thormählen, Benjamin Ward, and Philip H S Torr. VideoTrace : Rapid interactive scene modelling from video. *ACM Transactions on Graphics (TOG)*, 26(3):86, 2007.

[149] Carlos A Vanegas, Daniel G Aliaga, and Bedrich Benes. Building reconstruction using manhattan-world grammars. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 358–365, 2010.

[150] Carlos A Vanegas, Daniel G Aliaga, and Bedrich Benes. Automatic Extraction of Manhattan-World Building Masses from 3D Laser Range Scans. *IEEE Transactions on Visualization and Computer Graphics*, 18(10):1627–1637, 2012.

[151] Carlos A Vanegas, Daniel G Aliaga, Peter Wonka, Pascal Müller, Paul Waddell, and Benjamin Watson. Modeling the Appearance and Behavior of Urban Spaces. In *Eurographics State of the Art Reports*, pages 93–117, 2009.

[152] Vivek Verma, Rakesh Kumar, and Stephen Hsu. 3D Building Detection and Modeling from Aerial LIDAR Data. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2213–2220. IEEE, 2006.

[153] George Vosselman, Ben Gorte, George Sithole, and Tahir Rabbani. Recognising structure in laser scanner point clouds. *International Archives of Photogrametry, Remote Sensing and Spatial Information Sciences*, 46(8):33–38, 2004.

[154] Tomás Werner and Andrew Zisserman. Model selection for automated architectural reconstruction from multiple views. In *British Machine Vision Conference (BMVC)*, pages 53–62, 2002.

[155] Tomás Werner and Andrew Zisserman. New Techniques for Automated Architectural Reconstruction from Photographs. In *European Conference on Computer Vision (ECCV)*, pages 541–555, 2002.

[156] Changchang Wu, Sameer Agarwal, Brian Curless, and Steven M Seitz. Schematic surface reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1498–1505, 2012.

[157] Changchang Wu, Jan-Michael Frahm, and Marc Pollefeys. Detecting Large Repetitive Structures with Salient Boundaries. In *European Conference on Computer Vision (ECCV)*, pages 142–155, 2010.

[158] Jianxiong Xiao, Tian Fang, Ping Tan, Peng Zhao, Eyal Ofek, and Long Quan. Image-based Façade Modeling. *ACM Transactions on Graphics (TOG)*, 27(5):161, 2008.

[159] Jianxiong Xiao, Tian Fang, Peng Zhao, Maxime Lhuillier, and Long Quan. Image-based street-side city modeling. *ACM Transactions on Graphics (TOG)*, 28(5):114, 2009.

[160] Jianxiong Xiao and Yasutaka Furukawa. Reconstructing the World's Museums. In *European Conference on Computer Vision (ECCV)*, pages 668–681, 2012.

[161] Kai Xu, Hanlin Zheng, Hao Zhang, Daniel Cohen-Or, Ligang Liu, and Yueshan Xiong. Photo-inspired model-driven 3D object modeling. *ACM Transactions on Graphics (TOG)*, 30(4):80, 2011.

[162] Shin Yoshizawa, Alexander Belyaev, and Hans-Peter Seidel. Smoothing by Example: Mesh Denoising by Averaging with Similarity-based Weights. In *IEEE International Conference on Shape Modeling and Applications (SMI)*, page 9, 2006.

[163] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

[164] Qian Zheng, Andrei Sharf, Guowei Wan, Yangyan Li, Niloy J Mitra, Daniel Cohen-Or, and Baoquan Chen. Non-local scan consolidation for 3D urban scenes. *ACM Transactions on Graphics (TOG)*, 29(4):1, 2010.

[165] Jiejie Zhu, Liang Wang, Ruigang Yang, and James Davis. Fusion of time-of-flight depth and stereo for high accuracy depth maps. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

APPENDIX A

Curriculum Vitae

Personal

Name	Irene Reisner-Kollmann
Date of birth	May 30th, 1984
Email	irene.reisner@gmx.at
Languages	German, English

Education

since 2010/01	Ph.D. student in Computer Science, Doctoral College on Computational Perception, Vienna University of Technology, Austria Advisors: Prof. Werner Purgathofer and Prof. Markus Vincze
2006/10 - 2008/12	Master study at the Vienna University of Technology Major: Computer graphics and digital image processing Thesis: A user interface for photogrammetric reconstruction
2003/10 - 2006/06	Bachelor study at the University of Applied Sciences, Hagenberg, Upper Austria. Major: Media Technology and Design
1998/10 - 2003/06	Business School HAK Rohrbach, Austria

Work Experience

since 2010/01	Project Assistant at Vienna University of Technology, Austria
2008/10 - 2009/12	Researcher at VRVis Research Center, Vienna, Austria
2008/03 - 2008/09	Master's thesis at VRVis Research Center, Vienna, Austria
2008/03 - 2008/06	Tutor at Vienna University of Technology, Austria
2007/09 - 2007/10	Internship at VRVis Research Center, Vienna, Austria
2006/02 - 2006/07	Internship at Goelz & Schwarz GmbH, Munich, Germany

Reviewing

3DPMVT 2010, WSCG 2012

Talks

March 2012	Optimization of Shape Boundaries with Multi-modal Constraints. VRVis Forum, Vienna, Austria
June 2011	Reconstructing Piecewise-Planar Surfaces from Point Clouds and Images. Russian-Austrian Joint Seminar on Visual Computing
March 2011	Reconstructing Buildings as Textured Low Poly Meshes from Point Clouds and Images. VRVis Forum, Vienna, Austria
September 2008	A User Interface for Photogrammetric Reconstruction. VRVis Forum, Vienna, Austria (End talk of master's thesis)
April 2008	A User Interface for Photogrammetric Reconstruction. VRVis Forum, Vienna, Austria (Start talk of master's thesis)

Peer-Reviewed Publications

I. Reisner-Kollmann, A. L. Fuhrmann, and W. Purgathofer. Interactive reconstruction of industrial sites using parametric models. In Proceedings of 26th Spring Conference on Computer Graphics (SCCG 2010), pages 119-126. May 2010.

I. Reisner-Kollmann, A. Reichinger, and W. Purgathofer. 3D Camera Pose Estimation using Line Correspondences and 1D Homographies. In Advances in Visual Computing: 6th International Symposium on Visual Computing (ISVC 2010), pages 41-52. 2010.

I. Reisner-Kollmann, C. Luksch, and M. Schwärzler.Reconstructing Buildings as Textured Low Poly Meshes from Point Clouds and Images. In Eurographics 2011 - Short Papers, pages 17-20. April 2011.

I. Reisner-Kollmann and S. Maierhofer. Consolidation of Multiple Depth Maps. In IEEE Workshop on Consumer Depth Cameras for Computer Vision (CDC4CV 2011). November 2011.

I. Reisner-Kollmann and S. Maierhofer. Segmenting Multiple Range Images with Primitive Shapes. In Proceedings of 19th International Conference on Systems, Signals and Image Processing (IWSSIP 2012). April 2012.

I. Reisner-Kollmann, S. Maierhofer, and W. Purgathofer. Reconstructing Shape Boundaries with Multimodal Constraints. In Computers & Graphics, to appear. 2013.